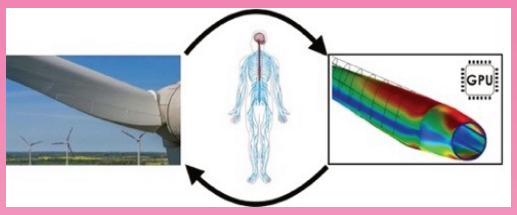
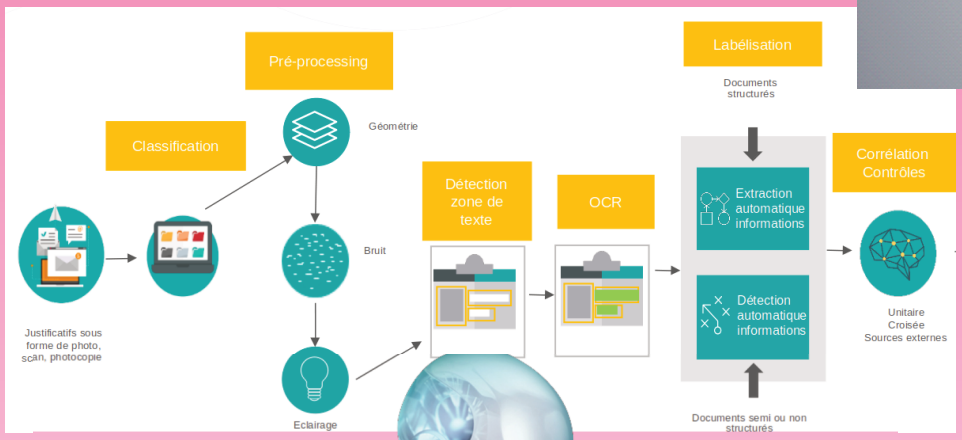
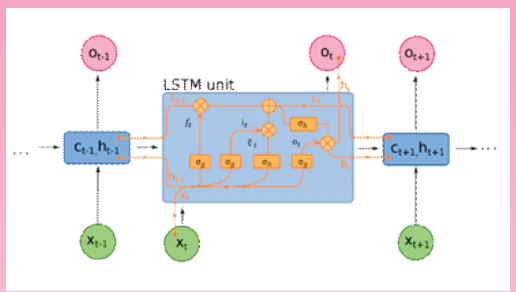
28ème année

La Revue 3E.I

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay







Intelligence Artificielle et Génie Électrique

2ème partie

Publication trimestrielle du Cercle Thématique 13.01 de la SEE

ENSEIGNER L'ÉLECTROTECHNIQUE ET L'ÉLECTRONIQUE INDUSTRIELLE

Société de l'Électricité, de l'Électronique et des Technologies de l'Information et de la Communication

see since 1883

N°109 – Juillet 2022

Abonnez-vous à La Revue 3E.I

- Outil didactique pour les filières préparant à l'enseignement technique
- Trame indispensable à la formation continue des hommes et des femmes de terrain
- 4 numéros par an : janvier, avril, juillet, octobre
- 3 dossiers complets par an sur des sujets d'actualité + 1 N° Pratique pédagogique

Choisissez votre formule d'abonnement pour 2022 :

❑ Version papier

4 numéros : janvier, avril, juillet, octobre.
Distribution postale

France & UE	Hors UE
❑ 40 € TTC	❑ 59,18 € HT*

❑ Version numérique

Accès aux publications numériques
ouvert pendant un an à compter de la date du paiement

France & UE	Hors UE
❑ 30 € TTC	❑ 29,38 € HT

❑ Version duo

Version imprimée
+ version numérique

France & UE	Hors UE
❑ 50 € TTC	❑ 68,97 € HT*

Votre adhésion à la SEE		
❑ Standard	❑ Retraité ❑ Enseignant ❑ Jeune actif (< 35 ans)	❑ Etudiant ❑ En recherche d'emploi
125 € TTC	65 € TTC	15 € TTC

+ Votre abonnement 3E.I *(Tarif réservé aux adhérents, version papier)*

❑ France & UE : 32 € TTC ❑ Hors UE : 51,34 € HT*

TVA de la revue 3E.I : 2,1 %. Adhésion collective possible via des conventions de partenariat - Contactez-nous à : sg@see.asso.fr
** Prix final incluant des frais de transports de 20 €*

Adresse de livraison

Nom* :

Prénom* :

Adresse* :

Code postal* : Pays* :

Ville* :

Tél.* :

e-mail* :

**Obligatoire*

Votre réglement

Je règle la somme de ____ €

par ❑ Chèque à l'ordre de la SEE
❑ Virement après réception de la facture
❑ Carte bancaire (Visa, Eurocard/Mastercard)

N° Carte

Date de validité ____ N° cryptogramme ____ *(3 derniers chiffres au dos de la carte)*

Adresse de facturation *(Si différente)*

❑ Je joins le bon de commande administratif N° ____ et je désire recevoir une facture au nom de mon employeur pour paiement à réception

Raison sociale de l'employeur :

Service : Activité (facultatif) :

Adresse :

Code postal : Ville :

Pays : N° TVA :

N° TVA intracommunautaire : obligatoire pour règlement HT en UE hors de France

e-mail* :

Date* ____ Signature* et cachet si il y a lieu :

**Obligatoire*

BULLETIN À COMPLÉTER ET RENVOYER À : SEE - 17 rue de l'Amiral Hamelin - 75116 Paris - France
Tél. +33(0)1 56 90 37 17 - abo@see.asso.fr

ABONNEMENT PLUS RAPIDE : www.see.asso.fr

see since 1883

❑ Je consens à recevoir les autres diffusions de la SEE & de ses activités (congrès, soirées débats, revues, etc.) qui sont extérieures aux diffusions liées à mon abonnement.

Conformément aux dispositions légales et réglementaires en matière de données personnelles, les informations recueillies sur ce formulaire sont enregistrées dans un fichier informatisé par la SEE (Société de l'électricité, de l'électronique et des technologies de l'information et de la communication) pour la mise en place et le suivi de l'abonnement souscrit ainsi que pour l'envoi de courriers, e-mails de réabonnements. Elles sont conservées et sont destinées à être utilisées par la SEE et les prestataires techniques de la SEE afin de permettre la bonne réception du magazine et d'assurer le service client. Vous pouvez exercer votre droit d'accès aux données vous concernant par courrier : SEE - Service abonnements 17 rue de l'Amiral Hamelin 75116 Paris ou par le formulaire de contact du site web : www.see.asso.fr. Offre valable du 01/10/2021 au 30/09/2022 dans la limite des quantités disponibles.

La Revue 3EI
publication trimestrielle de la ***SEE***

SOCIETE de l'ELECTRICITE, de l'ELECTRONIQUE et des TECHNOLOGIES de l'INFORMATION et de la COMMUNICATION.

17, rue de l'Amiral Hamelin, 75116 PARIS
Tél : 01 56 90 37 17
site web : www.see.asso.fr

SEE, association reconnue d'utilité publique par le décret du 7 décembre 1886
Siret 785 393 232 00042, APE 9412 Z, n° d'identification FR 44 785 393 232

3EI : Enseigner l'Electrotechnique et l'Electronique Industrielle

La Revue 3EI, Édition SEE,
17 rue de l'Amiral Hamelin
75116 PARIS

Directeur de la publication
François GERIN
Président de la SEE

Rédacteur en Chef
Franck LE GALL

Adresser les propositions d'article à :
revue3ei@gmail.com

Communication :
Mme. Mélisande DE LASSENCE
Communication1@see.asso.fr
01 56 90 37 17

Promotion et Abonnements :
(4 numéros par an)
Janvier, Avril, Juillet, Octobre.
Tél :01 56 90 37 17
abo@see.asso.fr

Tarifs 2022 :
Version PAPIER :
France et UE (TTC)..... 40.00 €
Pays hors UE (HT)..... 49,18 €
Version NUMERIQUE :
France et UE (TTC)..... 30.00 €
Pays hors UE (HT)..... 29,38 €
Version DUO (Papier+Num.) :
France et UE (TTC)..... 50.00 €
Pays hors UE (HT) ..... 58,97 €

Impression :
Dupliprint
733 rue Saint-Léonard 53100 Mayenne
Tel : 02 43 11 09 00
Couv :O.P. : All. – TFR :0 – C. : PEFC
Corp. :O.P. :Esp.– TFR :0 – C. : PEFC

Dépôt Légal : Juillet 2022

Commission Paritaire 1222 G 78028
ISSN 1252-770X

Sommaire du n° 109

p. 2 Éditorial,

Thème : Intelligence Artificielle et Génie Electrique

p. 3 Introduction à l'apprentissage par renforcement
Anthony Juton, Valentin Noël, Rida Lali

p. 16 Séries temporelles et réseaux de neurones récurrents
Valentin Noël

p. 22 Introduction aux bibliothèques Gym et Stable Baselines pour l'apprentissage par renforcement
Guénolé Chérot, Augustin Godinot

p. 25 Apprentissage par renforcement de la conduite d'un véhicule sur Airsim
Ludovic de Matteïs, Saša Radosavljevic

p. 31 Intégrer les connaissances physiques dans les réseaux de neurones : application à l'apprentissage de lois de comportement matériaux à partir de mesures de déformation par fibres optiques
Antoine Bénady, Ludovic Chamoin, Emmanuel Baranger

p. 34 Détection et classification automatique des documents pour l'application kyc
William Ketchantang, Dimitrios Tsolakidis, Kevin Meetooa

p. 39 Introduction aux méthodes d'accélération de réseaux de neurones
Edouard Yvinec Arnaud Dapogny Kévin Bailly

Hors Thème :

p. 48 Conception et développement d'une voiture autonome
Ayoub Karine, Maher Jridi, Rémi Adde et Sébastien Demouselle

p. 54 Conception, réalisation, et étalonnage d'une cellule d'efforts 6 axes à bas coût
Yan Barabinot, François Louf

p. 61 Montre connectée Projet pédagogique Bluetooth Low Energy, STM32, application smartphone Xamarin
Thomas Mongaillard

p. 71 Mutualisation des ressources IoT par conteneurisation de passerelle
Sylvain Lefebvre, Maher Jridi

p. 76 Challenge extrême défi « Véhicule faible impact » (ADEME)
A. Sivert, B. Vacossin, F. Betin, G. Plassat

Numéro 109 de la revue 3EI

C'est avec grand plaisir que nous sommes retrouvés à l'ENS Paris-Saclay pour la journée 3EI 2022 dédiée à l'enseignement du GE. Les présentations qui ont généré des échanges très fructueux sont maintenant disponibles sur le lien suivant : https://eduscol.education.fr/sti/si-ens-paris-saclay/actualites/journees-3ei-der-nikola-tesla

Le numéro que vous découvrez aujourd'hui fait suite à celui d'avril 2022. Il poursuit la réflexion sur « Intelligence Artificielle et Génie Electrique »

« Thème : IA et Génie Electrique : partie 2 »

Dans le numéro 108 de la revue 3EI, nous avons tenté de cerner l'intelligence artificielle, notamment l'IA non symbolique basée sur les réseaux de neurones. Les milliers de paramètres de ces réseaux sont optimisés grâce à l'apprentissage automatique. Par ailleurs, l'apprentissage supervisé à partir d'un jeu de données était lui aussi introduit.

Dans ce nouveau numéro (109), nous présentons une autre méthode d'apprentissage, basée sur des tests, nommée apprentissage par renforcement. Cette introduction à l'apprentissage par renforcement est accompagnée de deux articles proposant des travaux pratiques sur ce sujet autour des très puissantes bibliothèques Gym et Stable Baselines. Un dernier article sur les bases de l'IA présente les réseaux de neurones récurrents utilisés pour la prédiction.

Enfin, deux articles présentent des utilisations de l'IA dans le domaine de la recherche (modélisation de matériaux), et celui de l'industrie (traitement automatique de documents) et un dernier article s'attarde sur les méthodes de compression de réseaux de neurones pour accélérer leur traitement ou les implanter sur des systèmes embarqués peu puissants et peu gourmands.

« Hors Thème »

Les articles hors-thème sont principalement orientés vers des applications pédagogiques axées sur les projets.

L'équipe de l'ISEN Yncréa Ouest rassemblée autour de Maher Jridi nous propose le fruit d'un travail réalisé par des étudiants. L'objectif est de rendre autonome une maquette de voiture en permettant d'estimer la vitesse, de gérer le stationnement ou de détecter puis reconnaitre des panneaux de signalisation.

Yan Barabinot et François Louf de l'ENS Paris-Saclay présentent eux une solution très économe pour réaliser des TP avec une cellule 6 axes. Ils développent dans leur article la conception, la réalisation et un exemple d'utilisation d'une cellule 6 axes à faible coût utilisable dans le cadre de projets pédagogiques.

L'article de Thomas Mongaillard élève à l'ENS de Paris-Saclay, traite d'un projet de montre connectée qui permet la transmission des données de pulsation cardiaque vers un smartphone. Il étudie dans son article les différentes étapes permettant la mise en œuvre d'une connexion Bluetooth Low Energy entre un micro-contrôleur et un smartphone puis il détaille les étapes de la programmation du microcontrôleur et le développement d'une application Android avec l'outil Xamarin pour la réception des données sur smartphone.

Sylvain Lefebvre et Maher Jridi, nous décrivent quant à eux la plateforme FIT IOT Lab qui est une plateforme académique ouverte pour l'enseignement ou les expérimentations en IOT. Ce travail présente une méthode de réduction du nombre de composants matériels nécessaires à un déploiement.

Enfin, Arnaud Sivert et son équipe nous décrivent l'« Extrême défi » de l'ADEME qui nous lance le défi suivant : Comment produire des véhicules durables, équitables et viables à moindre coût ? Etes-vous tentés par l'aventure ?

Le Comité de Publication de la Revue 3EI

Faites connaitre notre revue
Vous en assurez la pérennité

INTRODUCTION A L'APPRENTISSAGE PAR RENFORCEMENT

ANTHONY JUTON[1], VALENTIN NOEL[2], RIDA LALI[3],

[1] : *Professeur agrégé au DER Nikola Tesla de l'ENS Paris Saclay, anthony.juton@ens-paris-saclay.fr*

[2] *Doctorant en imagerie médicale au laboratoire SATIE de l'ENS Paris Saclay, valentin.noel@ens-paris-saclay.fr*

[3] *Etudiant au DER d'informatique de l'ENS Paris Saclay, rida.lali@ens-paris-saclay.fr*

Résumé *: Cet article présente une méthode d'apprentissage de l'intelligence artificielle, bien adaptée à des problèmes pour lesquels il est possible de simuler le comportement du système dans son environnement (jeux vidéos, conduite autonome, asservissement de systèmes mécaniques...). Les données étant générées par l'interaction entre le système et son environnement, il n'est pas nécessaire de disposer d'un jeu de données comme pour l'apprentissage supervisé.*

Après avoir présenté l'apprentissage par renforcement, l'article développe en détail 2 algorithmes : le Q-Learning et le Deep Q-learning. S'intéressant au principe de l'apprentissage, il ne demande pas des connaissances en python très importantes.

Les codes commentés sont fournis sur Culture Sciences de l'Ingénieur sur le dépôt indiqué en [10] et sont un bon support pour qui voudrait adapter son propre système pour y faire de l'apprentissage par renforcement.

Mots clés *: apprentissage par renforcement, apprentissage par renforcement profond, reinforcement learning, Q-learning, Deep Q-learning, voiture autonome.*

I/ Introduction

L'apprentissage par renforcement (reinforcement learning RL) est une méthode d'apprentissage qui s'intéresse à la prise de décision.

Depuis l'état s (s comme state) de l'environnement, l'agent utilise une politique π pour choisir une action a. L'apprentissage par renforcement vise à optimiser la politique d'action π de l'agent grâce à un jeu de récompenses positives et négatives.

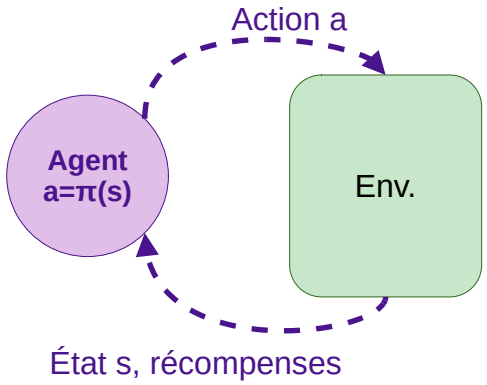


Figure 1 : Schéma d'interaction entre l'agent et son environnement

L'apprentissage demandant un grand nombre d'expérimentations, il est très utile voir indispensable de disposer d'un environnement simulé. Les jeux vidéos étant un environnement simulé avec des récompenses (le score), ils sont un support classique pour l'apprentissage par renforcement.

Le logiciel MuZero de DeepMind (société rachetée par Google), successeur d'Alpha Go, montre depuis 2018 le potentiel de l'apprentissage par renforcement, sans jeu de données préalable, pour être le meilleur au jeu de Go mais aussi à de nombreux jeux vidéos. [1]

Cet article, non exhaustif sur le sujet, présentera l'environnement des process markoviens dans lequel se place l'apprentissage par renforcement. Ensuite, il détaillera un des algorithmes d'apprentissage les plus simples : l'apprentissage tabulaire Q-learning. Enfin, nous introduirons l'apprentissage par renforcement profond où la politique d'action est issue d'un réseau de neurones, là encore à partir d'un des algorithmes les plus simples, le Deep Q-learning.

Trois autres articles de ce dossier fournissent des applications pour pratiquer les outils et algorithmes avancés de l'apprentissage par renforcement.

- *Stabilisation d'un pendule inversé à l'aide d'un apprentissage par renforcement* (revue 3EI n° 108 d'avril 2022), utilise Matlab
- *Introduction aux bibliothèques Gym et Stable Baselines pour l'apprentissage par renforcement* (ce numéro) utilise python et les très populaires bibliothèques gym et stable baselines.
- *Apprentissage par renforcement de la conduite d'un véhicule* (ce numéro) utilise python, gym, stable baselines et le simulateur de voitures autonomes réaliste AirSim.

II/ L'environnement

Avant de manipuler et de formaliser l'apprentissage par renforcement, il faut savoir identifier un problème adapté à l'apprentissage par renforcement pour sa résolution.

Comme décrit dans l'introduction, l'apprentissage par renforcement s'appuie sur l'interaction entre l'agent et son environnement.

L'**environnement** est d'abord dans un **état** s_t.

L'**agent** effectue une **action** a_t qui va avoir un impact sur l'environnement.

L'état de l'environnement est alors modifié (ou pas), il en résulte un nouvel état s_{t+1} ainsi qu'une **récompense** r_t qui permettra d'évaluer la pertinence de l'action choisie.

La fonction qui établit le passage de l'environnement de l'état s_t à l'état s_{t+1} sous l'effet de l'action a_t est nommée **fonction de transition**.

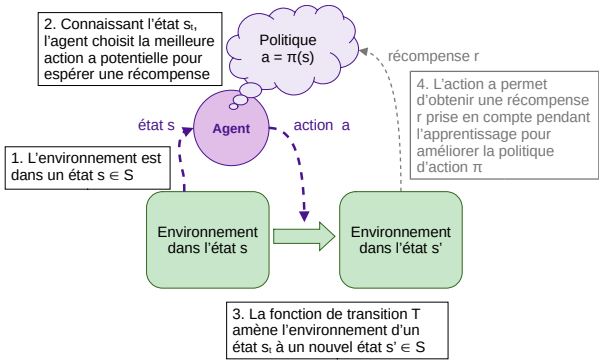


Figure 2: Principe de l'apprentissage par renforcement

L'optimisation de la politique ne se fait que pendant les périodes d'apprentissage. Une fois cet apprentissage jugé correct, il est possible d'utiliser la politique π pour agir sur l'environnement, sans chercher à l'améliorer.

II.1/ Le processus de décision markovien

L'apprentissage par renforcement se place dans le cadre des processus de décision markovien (Markov Decision Process MDP). L'évolution de l'environnement doit respecter la propriété de Markov : la fonction de transition T(s,a,s') doit être une fonction stochastique attribuant, pour une action a exécutée dans un état s, les probabilités de se retrouver dans chaque état s' et cette distribution de probabilité doit être indépendante des actions et état précédents.

Ainsi, le processus est modélisé par le quadruplet {S, A, T, R}

- **un ensemble d'états S**. ensemble (continu ou discret) des états de l'environnement que perçoit l'agent.
- **un ensemble d'actions A**, discret ou continu.
- **une fonction de transition T(s,a,s')** représentant la probabilité de se retrouver dans l'état s' en effectuant l'action a depuis l'état s.
- **une fonction de récompense R(s,a,s')**, indiquant la récompense obtenue lors du passage de s à s' en effectuant l'action a.

Notons que dans un **environnement déterministe**, la fonction de transition est simplifiée : elle indique directement l'état s' dans lequel se retrouve l'environnement après l'exécution de l'action a depuis l'état s.

Afin de mieux illustrer ce qu'est un processus de décision markovien, voici quelques exemples de problèmes adaptés à l'apprentissage par renforcement

II.2/ Exemple d'environnement

II.2.1/ Le pendule inversé

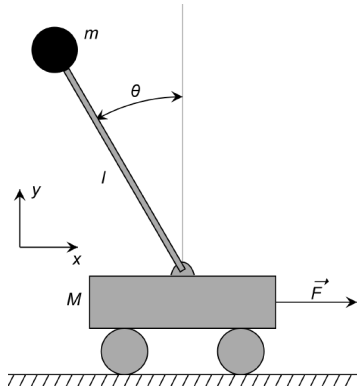


Figure 3: Schéma d'un pendule inversé

Le pendule inversé est un problème classique de contrôle optimal dont l'objectif est de maintenir à l'équilibre un poids sur un chariot mobile.

L'état est décrit par un vecteur composé de l'angle θ, de la vitesse angulaire ω, de la position relative du chariot ainsi que de la vitesse de ce dernier. L'ensemble S des états est donc un ensemble continu, que l'on peut si besoin discrétiser.

Les actions correspondent aux différentes valeurs de la force appliquée au chariot. C'est là encore un ensemble continu que l'on peut si besoin discrétiser.

En l'absence de frottement sec et si on se limite à de petits angles du pendule, la fonction de transition peut être modélisée par des équations linéaires indiquant la valeur suivante de l'état en fonction de l'état actuel et de l'action effectuée. On obtient alors un problème d'automatique classique. En présence de frottement sec et si on considère de plus larges déplacements angulaires du pendule, le problème est beaucoup plus complexe, mais reste déterministe. On pourrait alors écrire la **fonction de transition**, non linéaire.

Pour la **fonction de récompense**, on peut imaginer donner une récompense positive liée à l'angle que fait le pendule avec la verticale et une récompense négative en cas de chute du pendule (angle supérieur à 45° par exemple)

L'article *Stabilisation d'un pendule inversé à l'aide d'un apprentissage par renforcement* (revue 3EI n° 108 d'avril 2022) propose un exemple de séances de travaux pratiques autour de ce problème avec Matlab et un passage de la simulation à la réalité avec le système ControlX. L'article *Introduction aux bibliothèques Gym et Stable Baselines pour l'apprentissage par renforcement* de ce numéro propose d'utiliser lbibliothèque Gym pour l'apprentissage d'un pendule inversé (Cartpole).

II.2.2/ Corps articulé

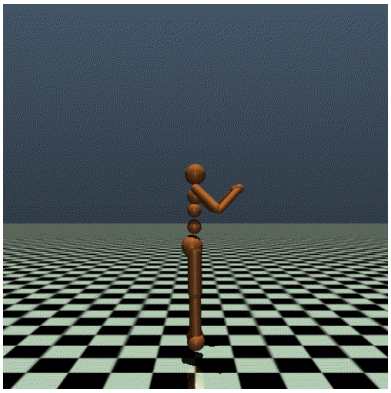


Figure 4: Simulation d'un humanoïde dans l'environnement Gym MuJoCo.
Image extraite de la documentation de Gym

Gym propose plusieurs problèmes alliant robotique, biomécanique et environnement 3D. Le but de celui présenté en Figure 4, l'un des plus ambitieux, est de maintenir un squelette mécanique debout et de le faire se déplacer sans le déséquilibrer.

L'état est constitué de l'angle et de la position relative de chaque articulation.

L'action est l'ensemble des couples appliqués à chaque articulation.

La fonction de transition peut prendre en compte les frottements secs de chaque articulation et donc être non linéaire. Elle peut aussi prendre en compte du bruit sur les commandes ou sur certains paramètres et des glissements stochastiques. Elle devient donc probabiliste.

La récompense est donnée par la combinaison des deux objectifs : être debout et se déplacer vers l'avant.

Un système physique réel étant infiniment complexe, pour passer du système réel au système simulé, il y a besoin d'expertise pour choisir les hypothèses que l'on peut faire pour simplifier le modèle sans être dans un cas trop simpliste. Ce choix est d'autant plus important, si on souhaite appliquer la politique apprise en simulation sur le système réel ensuite.

En parallèle des problèmes d'ingénierie pour lesquels on dispose ou on développe un simulateur, les jeux vidéos (ou les jeux ayant une variante vidéo) sont particulièrement propices au développement d'une IA entraînée par apprentissage par renforcement.

II.2.3/ Jeux vidéo Atari

Les jeux vidéos individuels fournissent un environnement informatique bien adapté à l'apprentissage par renforcement.

Les états sont les valeurs des pixels de l'écran de quelques images successives, **les actions** sont les actions possibles pour le joueur, **la fonction de transition** est fournie par le jeu (l'évolution de l'affichage à chaque action du joueur, qui doit respecter les propriétés du process de Markov), **les récompenses** sont les variations de score à chaque action (ou non action).

On pourrait imaginer des états de plus haut niveau (position de la balle, des briques et de raquettes pour un la casse-brique breakout par exemple).

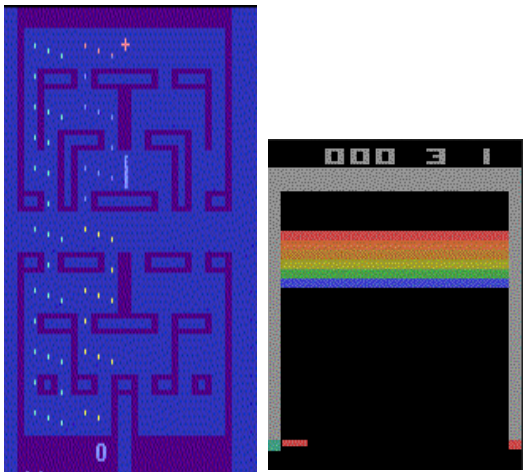


Figure 5: Jeux Atari Alien et Breakout, simulés à l'aide de l'environnement Gym.
Images extraites de la documentation de Gym

L'environnement Gym offre la possibilité de simuler différents jeux afin d'y implémenter des modèles d'apprentissage par renforcement. [9]

II.2.4/ Jeu à deux joueurs (Pong, Morpion, Go, Echec...)

Les jeux à deux joueurs, notamment les échecs et le go, demandent de simuler un adversaire pour l'apprentissage. Ils ont largement contribué aux succès médiatiques de l'IA : DeepBlue vainquit la champion Gary Kasparov aux échecs en 1997 et AlphaGo battit Lee Sedol au Go en 2017. Aucun joueur humain n'a aujourd'hui l'ambition de provoquer une nouvelle partie contre un de ces systèmes toujours plus performants.

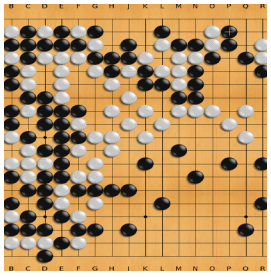


Figure 6: Jeu de go

En 1997, l'algorithme de DeepBlue, sans réseau de neurones, reposait sur le parcours dans un immense arbre. La programmation de DeepBlue avait requis les services d'un champion d'échec et les données de nombreuses parties.

Le Jeu de Go est un jeu d'une complexité telle qu'il ne peut être modélisé par un simple arbre de possibilité. En 2017, AlphaGo utilisait un réseau de neurones pour trouver un chemin dans cet arbre. Le réseau de neurones était entraîné au début pour imiter les meilleurs joueurs de go, à partir de leurs parties enregistrées (on peut parler d'apprentissage par imitation). L'apprentissage par renforcement prenait alors la suite.

Depuis 2018, AlphaGo Zero, renommé AlphaZero, puis MuZero apprennent avec succès les jeux de Go, Echecs, Shogi, Atari, sans aucune donnée préalable, avec des algorithmes issus de l'apprentissage par renforcement, en faisant des parties contre d'autres instances d'eux-même.

On peut alors imaginer le problème posé de la manière suivante : **les états** sont donnés par la position de toutes les pièces du jeu, **l'action** caractérise l'emplacement de la prochaine pièce, **la fonction de transition** est évidente, donnée par la règle du jeu, et **la récompense** est positive lorsque l'agent gagne la partie et négative s'il la perd.

Par exemple, il est possible en travaux pratiques avec des étudiants de modéliser un jeu de morpion (Tic-Tac-Toe en anglais). Ce jeu, bien plus simple que les jeux de Go ou d'échec, permettra de développer les premières intuitions concernant l'apprentissage à deux joueurs..

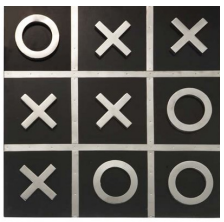


Figure 7: jeu de morpion

- **Les états** sont le contenu des 9 cases (soit 3^9= 19683 états possibles, sans a priori ni simplification),
- **Les 9 actions possibles** sont la case jouée.
- **La fonction de transition** est évidente (on ajoute la case jouée à l'état du jeu), si ce n'est qu'il faut y inclure le jeu de l'adversaire (et donc la case jouée par l'adversaire).
- **La fonction de récompense** peut donner 0,1 point négatif pour une action interdite (action sur une case déjà jouée), 1 point pour une victoire et -1 pour une défaite.

Pour commencer l'apprentissage de la politique de l'agent sur le jeu à 2 joueurs, il faut donner un algorithme (soit hasard complet, soit un peu intelligent) à l'adversaire de cet agent.

Ces exemples de problèmes présentés, intéressons-nous maintenant à l'apprentissage de la politique d'action.

III/ L'apprentissage

Une fois créé l'environnement {S,A,T,R} respectant les propriétés des process de Markov, il s'agit de mettre en place l'apprentissage par l'agent de sa politique π d'action, en essayant d'approcher le plus possible de la politique optimale notée π*.

Nous étudierons ici deux méthodes en détail, avec des exemples simples codés sans usage de framework, « from scratch » (les codes sont donnés sur le site Culture Sciences de l'ingénieur [10]). La première, une méthode tabulaire nommée Q-learning, relativement simple, permettra de bien comprendre les principes de l'apprentissage par renforcement. Ensuite, nous étudierons une méthode d'apprentissage profond où la politique d'actions est un réseau de neurones, ce qui permettra d'introduire les méthodes plus avancées d'apprentissage utilisées dans les autres articles sur l'apprentissage par renforcement.

IV/ Apprentissage tabulaire Q-learning

On s'intéresse ici au problème très classique du lac gelé (frozen lake – un esquimau cherche à rejoindre un igloo sur un lac gelé en évitant le trou), légèrement revisité pour tenir compte du changement climatique : un lapin cherche à rejoindre une carotte sur un pré glissant en évitant le renard.

A travers cet exemple, nous allons introduire les notations utilisées pour l'apprentissage par renforcement et l'algorithme de Q-learning utilisable pour les problèmes ayant un nombre limité d'états et d'actions. Il est possible de modifier le code pour l'utiliser pour résoudre un autre problème « markovien » à nombres d'états et actions limités.

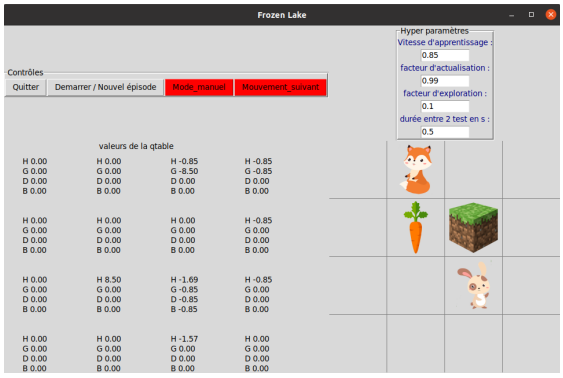


Figure 8: Environnement Frozen Lake revisité

Le projet comporte 2 fichiers, disponibles sur le dépôt indiqué en [10] :

- FrozenLake_2022_fenetre.py contient la description des fenêtres, des affichages et des boutons. Il n'est pas utile de l'ouvrir pour ce TP.
- FrozenLake_2022.py contient la description de l'environnement et l'apprentissage. Il reprend le formalisme utilisé par la populaire bibliothèque d'apprentissage Gym, sans l'utiliser.

L'environnement décrit dans la classe *Game* comprend un lapin qui, à partir de la case départ, doit atteindre la carotte sans sortir de la grille et en évitant le renard. Les blocs ne peuvent être franchis. L'agent est le contrôle du lapin.

Une fois la grille créée, les bloc, carotte et renard sont fixes. **L'état** (state) de l'environnement se limite donc à la position du lapin.

```
def _get_state(self):
    return self._position_to_id(*self.position_lapin)
```

Les actions sont les commandes de déplacement envoyées par l'agent au lapin : Haut, Gauche, Droite, Bas.

```
#Définition de l'environnement, états, récompenses, actions
class Game:
    ACTION_H = 0
    ACTION_G = 1
    ACTION_D = 2
    ACTION_B = 3
    #table des actions possibles
    ACTIONS = [ACTION_H, ACTION_G, ACTION_D, ACTION_B]
    #table des noms des actions
    ACTION_NAMES = ["H", "G", "D", "B"]
    #coordonnées des mouvements
    MOUVEMENTS = {
        ACTION_H: (0, -1),
        ACTION_G: (-1, 0),
        ACTION_D: (1, 0),
        ACTION_B: (0, 1)
    }
    #nombre d'actions possibles
    num_actions = len(ACTIONS)
```

La fonction de transition renvoie la nouvelle position du lapin si le déplacement était possible et la position actuelle sinon. Dans l'exemple suivi ici, la fonction de transition est déterministe : si la commande est Haut, le lapin va à la case du dessus, sauf si il n'y a pas de case ou si un bloc est sur cette case. Il est aussi possible d'activer la section en commentaire pour ajouter un peu d'aléatoire : l'herbe est alors mouillée et de façon aléatoire, le lapin glisse parfois et exécute alors une action proche de celle souhaitée.

La fonction de récompense attribue :

- -1 si la case de destination n'existe pas ou si elle est occupée par un bloc ou si elle est vide. On pénalise ainsi les déplacements inutiles.
- -10 si la case de destination est occupée par un renard.
- +10 si la case de destination est occupée par une carotte.

  Il est évidemment possible de développer une autre fonction de récompense, en ne pénalisant pas par exemple les déplacements inutiles.

La fonction move, appelée pour chaque action, comprend à la fois la fonction de transition (elle renvoie l'état suivant) et la fonction de récompense (elle renvoie la récompense) :

```
#fonction de déplacement : recoit une action et renvoie :
#le nouvel état de la grille, la récompense,
#si c'est terminé et des infos (0 ici)
#Il est possible d'y ajouter un peu d'aléatoire (les glissades du lapin)
def move(self, action):
    self.counter += 1

    if action not in self.ACTIONS:
        raise Exception("Invalid action")

    #ajout d'une action aléatoire :
    #le lapin glisse et ne fait pas ce qu'il aurait prévu
    # choice = random.random()
    #if choice < 0.1 :
    #    action = (action + 1) % 4
    #elif choice < 2 * 0.1 :
    #    action = (action - 1) % 4

    #calcul du déplacement prévu -> new_x et new_y
    d_x, d_y = self.MOUVEMENTS[action]
    x, y = self.position_lapin
    new_x, new_y = x + d_x, y + d_y

    #calcul des conséquences de l'action :
    #nouvelle position, récompense, fin du jeu, info(inutilisée)
    if self.block == (new_x, new_y):
        return self._get_state(), -1, False, 0
    elif self.renard == (new_x, new_y):
        self.position_lapin = new_x, new_y
        return self._get_state(), -10, True, 0
    elif self.carotte == (new_x, new_y):
        self.position_lapin = new_x, new_y
        return self._get_state(), 10, True, 0
    elif new_x >= self.n or new_y >= self.m or new_x < 0 or new_y < 0:
        return self._get_state(), -1, False, 0
    elif self.counter > 190:
        self.position_lapin = new_x, new_y
        return self._get_state(), -10, True, 0
    else:
        self.position_lapin = new_x, new_y
        return self._get_state(), -1, False, 0
```

On notera que la classe *Game* comprend également une fonction *generate_game* pour générer un environnement de manière aléatoire (position des blocs, renard, carotte) et une fonction *reset* pour remettre le lapin à sa position initiale.

On s'intéresse désormais à l'apprentissage permettant à l'agent d'optimiser sa politique d'actions et ainsi au lapin de trouver son chemin vers la carotte.

IV.1/ Principe du Q-learning

IV.1.1/ La Q-table

Le nombre d'états étant limité (16 ici) et le nombre d'actions étant limité également (4), il est possible de modéliser la politique d'action π par une table de qualité (**Q-table**) associant à chaque couple état-action une valeur de « qualité ».

	Action HAUT	Action GAUCHE	Action DROIT	Action BAS
État (0,0)	0.1	5.0	7.1	2.2
État (0,1)	-0.1	5.0	0.1	5.0
État (0,2)	-8.1	7.1	5.1	-4.0
...				
État (3,2)	0.1	5.0	0.1	5.0
État (3,3)	0.1	5.0	0.1	5.0

Figure 9: exemple de Q-table

Chaque case Q[s,a] représente la récompense totale (nommée aussi gain) espérée par l'agent si

- il démarre à l'état s
- effectue l'action a,
- applique ensuite la politique π

Ainsi, hors apprentissage, depuis l'état s, la politique d'action mènera à choisir l'action dont le gain espéré est maximal :

- $a = \pi(s) = \max Q[s,:]$

IV.1.2/ Algorithme d'apprentissage

L'algorithme d'apprentissage du Q-learning proposé par Sutton et Barto [2] est alors relativement simple. Il vise à s'approcher de la politique optimale pour maximiser la récompense cumulée. Le livre de Sutton et Barto ajoutera le formalisme mathématique pour les curieux.

Initialiser Q[s,a]
Répéter un nombre N de fois
    initialiser l'état s
    répéter
        choisir action a depuis s en utilisant Q et un peu d'aléatoire
        exécuter l'action a
        observer la récompense r et l'état s'
        mettre à jour Q :
        $Q[s, a] := Q[s, a] + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
        s := s'
        jusqu'à ce que s soit l'état terminal ou pour un nombre de boucles max

En regardant le code du fichier FrozenLake_2022.py, on reprend l'ensemble des étapes :

Initialiser Q[s,a]

```
#Création d'un nouvel environnement aléatoire
game = Game(4, 4)
game.print(espace_jeu)
Q = np.zeros([states_n, game.num_actions])
```

La Q-table créée est alors remplie de 0. Il peut être intéressant de remplir la table avec des valeurs aléatoires faibles, pour éviter de privilégier un comportement (celui des actions de faible indice) plutôt qu'un autre, lors des premières étapes d'apprentissage.

Répéter un nombre N de fois

```
#Démarrage de l'apprentissage sur un nombre d'épisodes fixé
for i in range(num_episodes):
```

Un épisode est un ensemble de pas (actions élémentaires) exécutées de l'initialisation à la fin du jeu (carotte ou renard atteint).

La condition d'arrêt de l'apprentissage est ici un nombre d'épisodes. On pourrait aussi choisir d'arrêter l'apprentissage quand il n'y a plus de progrès notable sur la récompense totale obtenue.

initialiser l'état s

```
#RAZ de l'environnement au début de chaque épisode
actions = []
s = game.reset()
states = [s]
cumul_reward = 0
fin_du_jeu = False
```

La fonction game.reset() remet le lapin à sa position initiale.

Répéter… jusqu'à ce que s soit l'état terminal ou pour un nombre de boucles max

```
#démarrage de l'apprentissage jusqu'à la fin de l'épisode (Renard ou Carotte trouvés)
while True:
```

Sur l'exemple limité Frozen Lake pour laquelle une fin de jeu (le lapin atteint la carotte ou le renard) n'est jamais loin, on peut ne faire se terminer la boucle que sur une fin du jeu :

```
if fin_du_jeu == True:
    break
```

choisir action a depuis s en utilisant Q et un peu d'aléatoire

```
# on choisit une action aléatoire avec une certaine probabilité, qui décroit petit à petit
#ou on choisit l'action permettant d'espérer la meilleure récompense
if random.random() < facteur_explo*(1. / (i/10 +1)) : #décroissance à régler
    a = random.randint(0,game.num_actions-1)
else :
    a = np.argmax(Q[s,:])
```

Dans l'état s, on remarque le choix de l'action a de manière aléatoire (exploration de nouvelles solutions) ou depuis la Q-table (exploitation des valeurs de la table déjà obtenues).

Le plus souvent, on choisit de diminuer l'exploration au fur et à mesure que s'améliore la Q-table. Le taux d'exploration et le taux de décroissance de l'exploration sont des hyper-paramètres importants pour la vitesse et l'efficacité de l'apprentissage. Trop d'exploration ralentit l'apprentissage et trop peu d'exploration risque de limiter Q à des premières solutions non optimales (minimum local).

exécuter l'action a

observer la récompense r et l'état s'

```
#obtention par l'environnement de l'état suivant et de la récompense
s1, reward, fin_du_jeu, _ = game.move(a)
```

La fonction game.move(a) exécute l'action a et renvoie le nouvel état (donné par la fonction de transition), la récompense (reward) et indique si le jeu est terminé (carotte ou renard atteint).

mettre à jour Q :

$Q[s, a] := Q[s, a] + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

$s := s'$

```
#Mise à jour de la table Q
Q[s, a] = Q[s, a] + lr*(reward + y * np.max(Q[s1,:]) - Q[s, a])
cumul_reward += reward
s = s1
```

L'essentiel du Q-learning est ici, dans la prise en compte de la nouvelle récompense pour améliorer Q. Q[s,a] s'approche de la récompense totale que l'on peut espérer en exécutant a depuis s. On y trouve donc la récompense obtenue en exécutant a (r) et la récompense totale maximale que l'on pourra espérer dans le nouvel état s' ($\max_{a'} Q(s', a')$).

On note l'introduction de 2 hyper-paramètres : **le taux d'apprentissage** α, nommé lr dans le code (pour learning rate) et **le facteur d'actualisation** γ, nommé y dans le code.

La nouvelle valeur de Q[s,a] est une somme de :

- l'ancienne valeur Q[s,a] multipliée par $(1-\alpha)$. On prend ainsi en compte l'ancienne valeur de

Q[s,a]. Plus le taux d'apprentissage (α ou lr) est élevé, plus on prend en compte la nouveauté et moins on prend en compte l'existant.

- la récompense (reward) obtenue en exécutant cette action. Elle est multipliée par le taux d'apprentissage (α ou lr), positif et strictement inférieur à 1.
- le maximum de récompense totale espéré dans l'état que l'on vient d'atteindre, multiplié par le facteur d'actualisation (γ ou y) et le taux d'apprentissage (α ou lr). C'est ce qui permet de propager les récompenses importantes vers les états permettant de les atteindre.

Une fois Q actualisée, on copie le nouvel état dans l'état actuel et on recommence.

Le programme permet ainsi d'observer, pas par pas, l'évolution de la Q-table.

Sur la Figure 10 est représentée la Q-table en fin d'apprentissage. La carotte étant en (0,0), on voit que l'action HAUT est très valorisée dans la case du dessous (1,0), de même que l'action Gauche dans la case de droite (0,1). On voit également que ces espérances de gain importantes ont « diffusé » vers les cases permettant de les atteindre (2,0), (1,1), (2,1), (2,2)… On peut voir l'inverse avec les actions menant vers le renard (en (1,3) et (2,3) par exemple).

Enfin, il est intéressant de voir que le taux d'exploration assez faible n'a pas permis de tester toutes les actions (l'action D en (0,1) n'a jamais été testée par exemple).

valeurs de la qtable

H 0.00 G 0.00 D 0.00 B 0.00	H -0.85 G 9.78 D 0.00 B 0.00	H 0.00 G 0.00 D 0.00 B 0.00	H 0.00 G 0.00 D 0.00 B 0.00
H 10.00 G 0.00 D 0.00 B 0.00	H 6.18 G -0.85 D -1.57 B -1.57	H -8.50 G -1.71 D -1.69 B -1.94	H 0.00 G 0.00 D 0.00 B 0.00
H 8.90 G 0.00 D 0.00 B 0.00	H -0.98 G 7.81 D -1.67 B -0.85	H -2.51 G 6.73 D -1.71 B -1.00	H -0.98 G -1.69 D -1.69 B -0.98
H 4.45 G 0.00 D 0.00 B 0.00	H -1.69 G -0.85 D -1.57 B -0.85	H -1.91 G -1.00 D -1.69 B -1.69	H -1.57 G -1.71 D -1.69 B -1.69

Figure 10: Exemple de Q-table en fin d'apprentissage

IV.2/ Retour sur les hyper-paramètres et autres méthodes tabulaires

La méthode Q-learning étant choisie, les performances de l'apprentissage (rapidité de convergence et caractère optimal de la politique trouvée) sont liés au choix des hyper-paramètres. On parle d'hyper-paramètres pour les distinguer des paramètres à optimiser que sont les valeurs de la Q-table. On retrouvera ensuite ces hyper-paramètres dans le deep Q-learning.

Un taux d'exploration (et un taux de décroissance de celui-ci pas trop élevé) permettent d'explorer plus de solutions et d'éviter les minimums locaux. En contre-partie, la convergence sera plus lente.

Un taux d'apprentissage (α ou lr) élevé permet de prendre en compte plus rapidement les nouvelles récompenses obtenues. Par contre, pour des problèmes complexes, il introduit de l'instabilité modifiant trop rapidement les paramètres.

Un taux d'actualisation (γ ou y) élevé donne plus de poids au gain espéré dans le nouvel état atteint par rapport à la récompense obtenue sur le moment.

Il est possible d'étudier l'influence de chaque hyper-paramètre indépendamment, même si le problème FrozenLake est peu significatif car trop simple.

On insère le code d'apprentissage dans une boucle sur plusieurs valeurs d'un hyper-paramètre. A chaque épisode, on stocke la récompense totale obtenue, puis une fois l'apprentissage terminée pour chaque boucle, on trace avec les évolutions de la récompense totale pour chaque valeur de l'hyper-paramètre grâce à la bibliothèque matplop

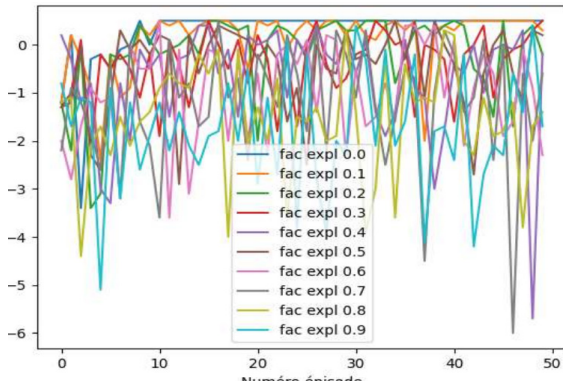


Figure 11: Exemple d'étude de l'influence d'un hyper-paramètre : Cumul des récompenses en fonction du facteur d'exploration

Avant d'aborder l'apprentissage par renforcement profond, notons que d'autres méthodes tabulaires existent. SARSA par exemple utilise également une Q-table mais sa mise à jour pendant l'apprentissage est un peu différente du Q-learning.

V/ Apprentissage par renforcement profond

V.1/ Q-apprentissage profond : définition et intérêt

V.1.1/ Du Q-apprentissage traditionnel au Q-apprentissage profond.

Pour la plupart des problèmes ayant un environnement possédant un nombre limité d'actions et d'états possibles comme Frozen Lake présenté ci-dessus, le Q-apprentissage « tabulaire » permet d'afficher la Q-fonction sous forme de table. Pour des problèmes possédant un très large nombre d'actions et d'états possibles, il est bien plus coûteux informatiquement, voir impossible, de représenter la Q-fonction dans une table. En effet, dans un environnement comportant n_s états et n_a actions, la table comporte ainsi n_s*n_a cellules ce qui peut, pour des valeurs n_s et/ou n_a élevées, entraîner des coûts computationnels élevés. Ces coûts sont divisés en deux parties, la première étant la quantité de mémoire nécessaire pour sauvegarder et mettre à jour la table qui augmenterait avec le nombre d'états, la deuxième étant le temps nécessaire à l'exploration de chaque état pour créer la Q-table requise.

V.1.2/ Approximation des valeurs de Q par réseau de neurones.

Au lieu de représenter la fonction Q par une table, il est possible de la représenter par une fonction continu. Cette fonction servira d'approximation. N'importe quel type de fonction peut fonctionner : des polynômes, des arbres de décisions… Les réseaux de neurones sont également de bons candidats. Il a ainsi été proposé d'approximer la Q-fonction par le biais de réseaux de neurones comportant des paramètres W (W représente l'ensemble des poids et biais du réseau de neurones), soit Q(s,a;W) ≈ Q*(s,a), Q* correspondant à la Q-fonction optimale. Un tel réseau de neurones est appelé Q-réseau et prend en entrée l'état et retourne en sortie la valeur de Q de toutes les actions possibles, comme illustré dans la *Figure 12*.

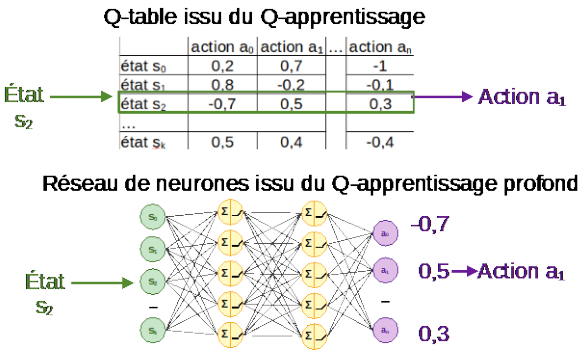


Figure 12: Q-apprentissage vs. Q-apprentissage profond [5]

V.2/ Défis de l'apprentissage profond par renforcement par rapport à l'apprentissage profond.

Nous nous intéressons dans cette section aux différentes étapes de l'apprentissage par renforcement avec l'utilisation d'un réseau de Q-apprentissage profond (deep Q-learning) et aux différences dans l'apprentissage amenées par la nature même de l'apprentissage par renforcement.

En effet, comme présenté dans l'article *Introduction à l'apprentissage profond* (revue 3EI numéro 108), l'entraînement supervisé d'un réseau de neurones se fait via un jeu de données déjà étiquetées. Dans l'apprentissage par renforcement, il n'y a pas de jeu de données étiquetées, l'entraînement se fait par les expériences. C'est d'ailleurs un des intérêts de l'apprentissage par renforcement, tout un chacun pouvant travailler sur son problème de type process de Markov, sans besoin d'avoir un énorme jeu de données déjà disponible et donc il n'y a pas d'obligation de se limiter aux problèmes classiques où des jeux de données publics existent.

Cet entraînement par les expériences induit quelques modifications dans l'apprentissage qui seront détaillées ici.

V.2.1/ Apprentissage par renforcement par le biais de réseaux de Q-apprentissage profond

A l'instar d'un réseau de neurones profond, un réseau de neurones de Q-apprentissage profond (cf. Q-réseau) comporte une ou plusieurs entrée(s) et sortie(s), des poids, une fonction de coût, etc. Leurs architectures et fonctionnement sont ainsi similaires.

Dans la partie précédente, le Q-learning utilisait à chaque pas l'équation suivante pour approcher la Q-table de la Q-table optimale :

$Q[s, a] := Q[s, a] + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

Q[s,a] s'approchait ainsi avec le taux d'apprentissage α de la valeur de récompense totale maximale espérée :

- **r** obtenue en exécutant l'action
- $\gamma \max_{a'} Q(s', a')$, récompense maximale que l'on pourrait obtenir dans l'état suivant

  En apprentissage profond, on cherche aussi à approcher de la récompense maximale espérée $r + \gamma \max_{a'} Q(s', a'; W)$, d'une manière assez semblable : on va chercher à minimiser, à chaque étape i, une fonction coût L_i égale à :

$$L_i(W_i) = \left[\left(r + \gamma \max_a Q(s', a'; W_i) - Q(s, a; W_i)\right)^2\right]$$

On utilise alors les fonctions d'optimisation présentée dans l'article sur l'apprentissage profond.

A l'instar d'autres réseaux de neurones, l'apprentissage du Q-réseau peut utiliser la rétropropagation du gradient, une méthode consistant à mettre à jour les poids de chaque neurone de la dernière couche vers la première en fonction de l'erreur déterminée par la fonction de coût L_i.

La prise en compte de la modification, à chaque pas, se fait avec un taux d'apprentissage. Si celui-ci est faible, chaque pas apporte une modification mineure aux valeurs du réseau de neurones. On évolue doucement mais on

évite des oscillations brutales autour des valeurs intéressantes.

V.2.2/ Le défi du renforcement par apprentissage profond : la nécessité d'un second réseau de neurones

Contrairement à l'apprentissage profond, où la cible (les données étiquetées) est stationnaire menant à un apprentissage stable, nous avons donc une cible (dans **r + γ max$_{a'}$Q(s',a';W))** non-stationnaire dans le cadre du renforcement par apprentissage profond. Nous sommes donc dans une situation où le Q-réseau calcule à la fois la valeur prédite et la valeur cible, ce qui peut entraîner beaucoup de divergence entre ces deux valeurs. De cette observation est née l'idée d'utiliser deux réseaux de neurones, le Q-réseau et un réseau se chargeant du calcul de la valeur cible, que nous appelons ici réseau cible. Le réseau cible possède la même architecture que le Q-réseau mais avec des paramètres fixes, en pratique, un instantané des paramètres réseau d'il y a quelques itérations au lieu de la dernière itération.

En d'autres termes, un hyper-paramètre C est défini, de telle sorte que toutes les C itérations les paramètres du Q-réseau soient copiés et deviennent les nouveaux paramètres du réseau cible comme illustré dans la Figure 13.

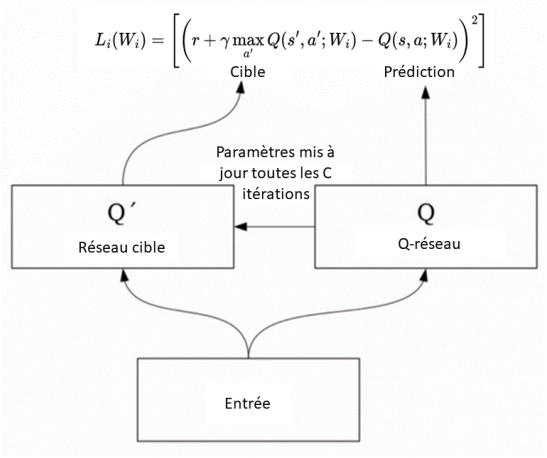


Figure 13: Architecture et fonctionnement de l'approche par deux réseaux de neurones [7]

De ce fait, la fonction cible reste fixe entre chaque C itérations, menant à un apprentissage plus stable.

Dans le cas d'un apprentissage par renforcement par le biais du Q-réseau nommé Q, on retrouve les trois étapes du Q-learning plus la mise à jour du réseau cible Q_{cible} :

1. Dans un état donné, la prochaine action qui sera effectuée est celle ayant la plus grande valeur en sortie du Q-réseau, ou une action aléatoire (exploration qui diminue avec le temps)
2. La fonction de coût est l'erreur moyenne carrée de la différence entre la valeur de Q_{cible} et la valeur de cible.
3. On optimise le réseau Q avec la rétropropagation du gradient
4. Chaque nombre de pas fixé, on recopie Q dans Q_{cible}.

En d'autres termes, en reprenant l'approximation effectuée en V.1.2 :

$$Q(s,a;W) \approx Q^*(s,a)$$

Elle-même étant issue de l'idée de base du Q-apprentissage, c'est-à-dire l'équation d'optimalité de Bellman comme mise à jour itérative à chaque étape i.

De cette équation, on peut formuler la fonction de coût L_i du Q-réseau à chaque étape i (Q_{cible} y est noté Q'):

$$L_i(W_i) = \left[\left(r + \gamma \max_{a'} Q'(s', a'; W_i) - Q(s, a; W_i)\right)^2\right]$$

VI/ Exemple pratique de la voiture autonome

VI.1/ Présentation de l'environnement

L'exemple Frozen Lake étant trop simple pour mettre en valeur l'apprentissage profond, on propose un second problème, la voiture autonome, avec un modèle physique extrêmement simple permettant des apprentissages en 3 à 15 mn suivant la machine et l'algorithme d'apprentissage.

La voiture dispose de 3 capteurs type télémètres et de 3 actions possibles (à gauche de 3 degrés, tout droit, à droite de 3 degrés). Chaque capteur donne une valeur normalisée entre 1/50 et 1 correspondant à la distance de l'obstacle, en pixels/50.

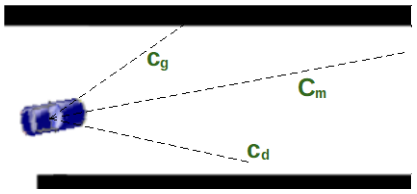


Figure 14: Description de l'environnement voiture autonome

La fenêtre propose une piste d'entraînement et une piste de validation, pour détecter le sur-apprentissage (cas où la voiture apprend trop bien la piste d'entraînement et ne peut plus généraliser à une autre piste).

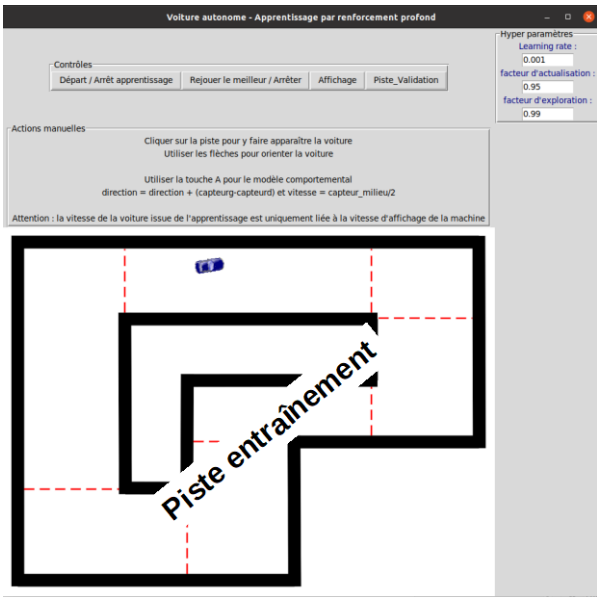


Figure 15: fenêtre d'apprentissage avec la piste d'entraînement. Les pointillés rouge, invisibles de la voiture, sont les balises attribuant +100 de récompense.

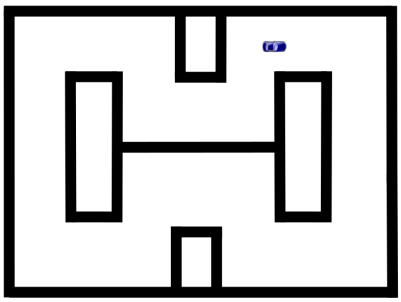


Figure 16: piste de validation

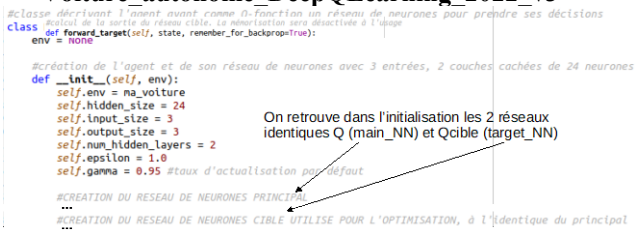
Les fichiers sont déposés sur le dépôt [10] :

Voiture_autonome_2022_fenetre_v2.py décrit les fenêtres et boutons, il n'est pas utile de l'ouvrir pour l'exercice. La touche 'A' donne un exemple d'IA symbolique, avec un algorithme comportemental.

Voiture_autonome_2022_voiture_v2.py décrit la classe voiture dans lequel on trouve l'initialisation, les déplacement, les récompenses. La méthode reset de la classe voiture positionne la voiture de manière aléatoire dans une zone de départ pour éviter le sur-apprentissage.

Voiture_autonome_Q_learning_2022_v2 présente un apprentissage par renforcement de type Q_learning, équivalent à celui présenté précédemment, mais avec 50^3 = 125000 états et donc une table de 375000 valeurs {état,action} !

Voiture_autonome_DeepQLearning_2022_v3



présente un apprentissage par renforcement profond.

Notons que pour l'affichage de l'image de la voiture avec un angle quelconque, il faut la bibliothèque PIL :

```
sudo apt install python3-pil python3-pil.imagetk
```

VI.2/ Le réseau de neurones de l'agent de conduite

C'est donc à ce dernier fichier, celui de l'apprentissage profond, que l'on s'intéresse, en soulignant les éléments importants du code.

La Q-fonction de conduite de la voiture associe une valeur aux 3 actions à chaque état du système (les valeurs de ses 3 capteurs). En exploitation, on choisit l'action ayant la plus grande valeur.

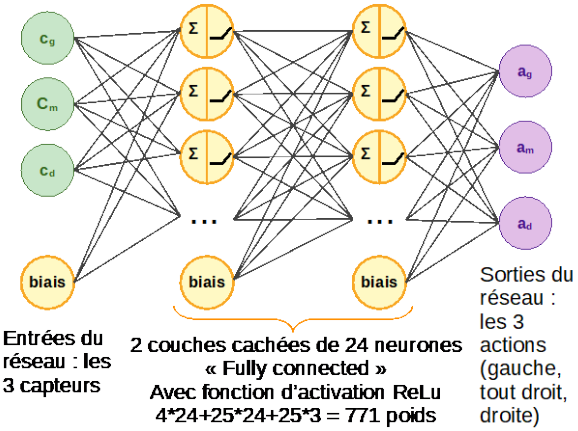


Figure 17: Réseau de neurones de la Q-fonction « conducteur »

Voiture_autonome_DeepQLearning_2022_v3 contient :

- **La fonction d'activation** et sa dérivée

```
def relu(mat):
    return np.multiply(mat,(mat>0))

def relu_derivative(mat):
    return (mat>0)*1
```

- **La classe NNLayer** pour une couche de neurones, avec les méthodes forward et backward, méthode qui prend en compte la descente de gradient, avec le taux d'apprentissage (learning rate)

```
# classe décrivant une couche de réseau de neurones
class NNLayer:
    # initialisation : nombres d'entrées, nombre de neurones (sorties), fonction d'activation
    def __init__(self, input_size, output_size, activation=None, lr = 0.001):
        self.input_size = input_size
        self.output_size = output_size
        #les poids sont initialisés avec une valeur aléatoire
        self.weights = np.random.uniform(low=-0.5, high=0.5, size=(input_size, output_size))
        self.activation_function = activation
        self.lr = lr
# Calcul des sorties d'une couche à partir des entrées (mode "forward")
def forward(self, inputs, remember_for_backprop=True):
#mise à jour des poids avec le gradient, pondéré par le taux d'apprentissage
def update_weights(self, gradient):
    self.weights = self.weights - self.lr*gradient
#mise à jour de la couche de neurones à partir du gradient venant de la couche suivante
def backward(self, gradient_from_above):
```

- **La classe RLAgent** est celle dont l'instance, nommée model, sera l'agent de notre problème, chargé de prendre les décisions à partir d'un réseau de neurones et de l'optimiser.

```
#classe décrivant l'agent avant comme Q-fonction un réseau de neurones pour prendre ses décisions
class RLAgent:
    def forward_target(self, state, remember_for_backprop=True):
        env = None

    #création de l'agent et de son réseau de neurones avec 3 entrées, 2 couches cachées de 24 neurones
    def __init__(self, env):
        self.env = ma_voiture
        self.hidden_size = 24
        self.input_size = 3
        self.output_size = 3
        self.num_hidden_layers = 2
        self.epsilon = 1.0
        self.gamma = 0.95 #taux d'actualisation par défaut

        #CREATION DU RESEAU DE NEURONES PRINCIPAL
        ...
        #CREATION DU RESEAU DE NEURONES CIBLE UTILISE POUR L'OPTIMISATION, à l'identique du principal
        ...
```

On retrouve dans l'initialisation les 2 réseaux identiques Q (main_NN) et Qcible (target_NN)

- La méthode select_action est celle qui permet d'obtenir a depuis s. Le taux d'exploration est réglé par l'attribut epsilon, décroissant au fil des pas.

```
#exploration ou application de la politique pour choisir une action
def select_action(self, state):
```

- La méthode train de la classe RLAgent gère l'apprentissage. Elle utilise les 2 méthodes forward (pour le réseau principal et le réseau cible) pour les calculs des sorties connaissant les entrées, la méthode backward pour l'optimisation des paramètres par rétropopagation de gradient.

```
#entrainement à partir de a, s',s, r : calcul des valeurs d'action données par Q
def train(self, done, action, new_state, state, reward):
    action_values = self.forward(state, remember_for_backprop=True)
    next_action_values = self.forward_target(new_state, remember_for_backprop=False)
    experimental_values = np.copy(action_values)
    #application de la formule du deep Q-learning avec -100 si crash ou reward si
    if done:
            experimental_values[action] = -100
    else:
            experimental_values[action] = reward + self.gamma*np.max(next_action_
    #Mise à jour des poids par la propagation du gradient vers les couches amont
    self.backward(action_values, experimental_values)
#calcul de la sortie du réseau de neurones en calculant couche après couche,
def forward(self, state, remember_for_backprop=True):
#calcul de la sortie du réseau cible. La mémorisation sera désactivée à l'usage
def forward_target(self, state, remember_for_backprop=True):
#Propagagation du gradient vers les couches amont, mise à jour des poids
def backward(self, calculated_values, target_values):
```

On y retrouve la prise en compte de la récompense et de la récompense maximale de l'état suivant, avec le facteur d'actualisation.

VI.3/ L'entraînement

```
# The main program loop
for i_episode in range(NUM_EPISODES):
    #positionnement aléatoire de la voiture au départ
    ma_voiture.reset()
    state = ma_voiture.lecture_capteurs(fenetre.image_piste)
    fenetre.maj_affichage()
    cumul_reward = 0
    index_recopie_target_NN = 0

    # On commence un pas
    while True :
        index_recopie_target_NN +=1
        #on choisit une action en utilisant la politique ou l'exploration (c'est pris
        action = model.select_action(state)
        #on fait un pas et on cumule la récompense
        new_state, reward, done, info = ma_voiture.step(action,fenetre.image_piste)
        cumul_reward += reward

        # On entraine le réseau avec ce pas
        model.train(done, action, new_state, state, reward)
        state = new_state

        #A chaque nombre de pas fixé, on recopie le réseau principal dans le réseau cible
        if (index_recopie_target_NN %20 == 0) : #mise à jour de la NN_target
            for i in range(model.num_hidden_layers+1) :
                model.target_NN[i].weights = np.copy(model.main_NN[i].weights)

        if done:
            #on met à jour la meilleure récompense total et le meilleur réseau
            if cumul_reward > cumul_reward_best :
                for i in range(model.num_hidden_layers+1) :
                    model_best.main_NN[i].weights = np.copy(model.main_NN[i].weights)
                cumul_reward_best = cumul_reward
            break
```

initialisations

Choix d'une action, avec exploration (intégrée par *select_action*)

Exécution d'un pas

Optimisation du réseau

Chaque 20 pas, on met à jour Qcible

Si la récompense est très bonne, on garde une copie du réseau.

L'algorithme de l'entraînement ressemble beaucoup à celui du Q-learning.Validation - Exploitation de l'inférence

Une fois que l'apprentissage paraît satisfaisant, arrêter l'apprentissage puis lancer via le bouton Rejouer l'exploitation du meilleur réseau.

```
#gestion du rejeu de l'inférence
if fenetre.rejouer :
    fenetre.rejouer= False

    state ,_,_,_ = ma_voiture.step(0,fenetre.image_piste)
    while(True): #tant qu'il n'y a pas d'accident
        action = model_best.select_action(state)
        state ,_,_,_ = ma_voiture.step(action,fenetre.image_piste)
        fenetre.maj_affichage()
        if(state[1]<0.02) : #en cas d'accident on arrête la simulation
            break
        if fenetre.start == True :
            fenetre.start = False
            break
        if fenetre.rejouer :
            fenetre.rejouer= False
            break
```

Il est possible de placer avec la souris la voiture n'importe où sur la piste d'entraînement comme sur la piste de validation.

Les hyper-paramètres choisis pour le Q-learning notamment conduisent à une table Q qui permet de tourner sur la piste d'entraînement mais pas sur la piste de validation (un demi-tour maximum).

VI.4/ Exercices envisagés

L'environnement maîtrisé, il est possible d'affiner le système en augmentant :

- le nombre d'actions possibles (qui restent discrètes),
- le nombre de télémètres (on peut se rapprocher d'un Lidar à 200 faisceaux),
- la résolution de ces télémètres…

On pourrait aussi envisager d'améliorer la physique de la voiture (adhérence, inertie, rotation type « bicyclette »…), avec le risque d'augmenter le temps de calcul. Pour plus de réalisme, on peut consulter l'article *Apprentissage par renforcement de la conduite d'un véhicule* de ce numéro.

On peut également imaginer faire courir 2 voitures sur la même piste. On peut faire de nouvelles pistes (fichier .png noir & blanc)

Un autre aspect intéressant, notamment dans une classe composée de nombreux étudiants-expérimentateurs, est d'étudier l'influence des hyper-paramètres en traçant l'évolution de la récompense cumulée pour différentes variations de chaque hyper-paramètre :

- taux d'apprentissage
- facteur d'actualisation
- facteur d'exploration,
- nombre de pas rythmant la mise à jour du réseau cible,
- vitesse à laquelle diminue le taux d'exploration,
- nombre de neurones par couche et nombre de couches,
- les récompenses utilisées.

Notons qu'il est simple de modifier la méthode *reset* de la classe voiture pour supprimer le positionnement aléatoire au démarrage et ainsi mettre en évidence le sur-apprentissage (la voiture ne fonctionne qu'en partant de ce point).

VII/ Influence de l'environnement sur l'apprentissage par renforcement profond

Maintenant que nous avons clairement défini l'architecture et le fonctionnement de nos deux réseaux de neurones, il est d'intérêt de se pencher sur l'impact de l'environnement sur l'apprentissage de notre agent. En effet, nous avons ici particulièrement présenté deux problèmes, Frozen Lake et la voiture autonome. Leurs environnements sont très différents ainsi que l'apprentissage dans chacun de ces cas. Le principal aspect que nous voulons mettre en avant est la différence entre ces environnements et la similarité de leurs états interdépendants. Ce sujet est particulièrement important car directement corrélé à l'évolution de la complexité de l'environnement des problèmes étudiés et donc principalement présent dans les problèmes d'apprentissage par renforcement profond.

VII.1/ Similarité des états interdépendants

Pour illustrer ce que nous appelons ici la similarité des états interdépendants, nous utilisons de nouveau l'exemple de la voiture autonome. L'objectif est de lui apprendre à conduire sur un circuit. Imaginons que le début du circuit soit une longue ligne droite. Dans ce cas, l'agent n'aura rien appris pour la gestion de courbes. Contrairement, un circuit totalement circulaire n'apprendra pas à notre agent à se déplacer en ligne droite. De ce fait, si les agents cités au-dessus sont déposés sur le circuit de l'autre, ils ne pourraient aucunement le parcourir. Pour contourner cet effet de l'apprentissage par renforcement dit « en ligne » (l'agent peut interagir comme bon lui semble avec l'environnement), nous nous tournons vers l'apprentissage par renforcement dit « hors ligne ». Ces deux types d'apprentissage sont illustrés dans la *Figure 18*.

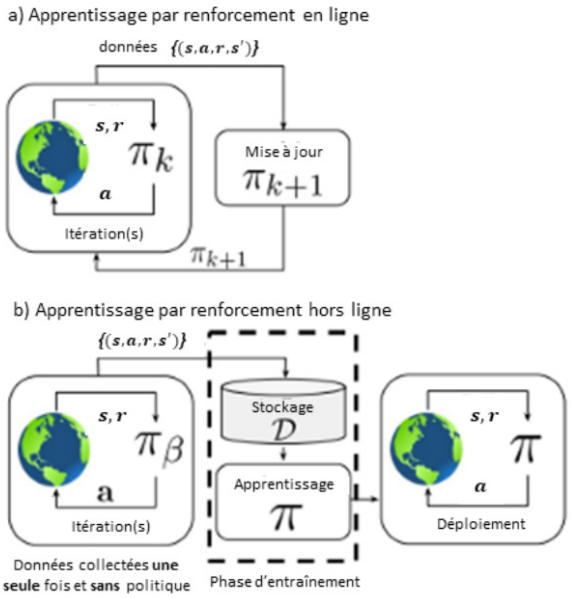


Figure 18: Principes de fonctionnement : a) l'apprentissage par renforcement en ligne, b) l'apprentissage par renforcement hors ligne [8], π_β est juste un embryon de politique.

VII.2/ Apprentissage par renforcement hors ligne

L'apprentissage par renforcement purement hors ligne empêche l'agent d'interagir avec l'environnement. C'est utile par exemple pour travailler avec des données réelles de véhicule issues d'essais avec un conducteur (qui applique sa propre politique π_β). Le conducteur essaiera alors de passer dans le plus d'états possibles.

L'agent apprend donc maintenant depuis un lot d'expériences, qui est aléatoirement échantillonné, duquel l'agent sélectionne un échantillon uniformément distribué et effectue son apprentissage grâce à lui.

L'échantillonnage aléatoire des expériences permet ainsi de réduire largement le biais introduit par la possible nature séquentielle d'un environnement, représenté par l'exemple de la ligne droite.

VII.3/ Mémorisation pour l'apprentissage par renforcement hybride

Un autre méthode utilisée pour l'apprentissage par renforcement, nommée experience replay, est hybride entre apprentissage en ligne et hors-ligne.

Comme nous l'avons vu, il n'est pas souhaitable d'optimiser le réseau Q en utilisant pour cible des données issues de ce réseau Q.

Une alternative à l'utilisation d'un réseau cible est l'utilisation d'expériences mémorisées pour l'optimisation.

A chaque essai, on enregistre l'expérience (s, a, r, s') et on fait l'optimisation sur des données pris aléatoirement dans la base de données d'expériences.

Un exemple commenté d'apprentissage par renforcement profond avec experience replay est présenté dans le fichier :

Voiture_autonome_DeepQLearning_2022_experience_replay.py

Conclusion

Dans cet article ont été mis en avant les fondements de l'apprentissage par renforcement et de l'apprentissage par renforcement profond. Des exemples généralistes et largement étudiés ont été utilisés pour illustrer les propos tenus. Cependant, nous ne pouvons que souligner la non exhaustivité de la liste des applications possibles montrées ici. De la même façon, les algorithmes mis en avant dans cet article sont nécessaires à la compréhension des premiers pas à la fois de l'apprentissage par renforcement tabulaire et profond.

De nombreux autres algorithmes et approches ont ainsi émergés, outrepassant certaines limites inhérentes aux approches présentées. Pour n'en citer que les plus fameuses d'entre elles, le Deep Q-Network (DQN) est l'évolution direct du Q-Apprentissage Profond étudié, le Proximal Policy Optimization (PPO), qui se décline en 2 variantes majeures qui toutes les deux offrent en règle générale de meilleures performances et une meilleure convergence. Cette méthode est cependant peu robuste car sensible aux changements. Dans un autre registre, le Soft-Actor Critic (SAC) est très efficace pour les techniques d'optimisation basées sur l'énergie en raison de la régularisation du facteur entropique. En d'autres termes, SAC utilise la régularisation entropique, la politique étant entraînée à maximiser un compromis entre la valeur de Q prédite et l'entropie. Ces trois approches correspondent aux méthodes principalement utilisées dans l'état de l'art actuel et leur intérêt pratique étant évident, il est conseillé de s'intéresser à leur fonctionnement, par exemple en s'aidant des ressources disponibles en [9].

Ecrire les optimisations des réseaux de neurones a permis de bien comprendre leur fonctionnement. Pour le travail sur des problèmes importants, avec les algorithmes DQN, PPO ou SAC, StableBaseLine est un outil très populaire. Il est présenté dans un autre article, avec l'environnement Gym. Un troisième article utilise Gym, StableBaseLine et le simulateur AirSim pour un simulateur de voitures autonomes réaliste.

Enfin, on a considéré ici l'observation o de l'environnement par l'agent comme étant son état (s = o). Cependant, dans certains travaux, les deux sont distingués. En effet, dans certains types d'apprentissage par renforcement plus complexes, l'état est « partiellement observable ». On parle de processus de Markov partiellement observables (POMDP). Ainsi, on essaiera de déduire l'état à travers les observations générées. C'est le cas notamment à travers les travaux sur l'Inférence Active mené par Karl Friston [8] (qui consiste à caractériser les comportements des individus via leurs émotions et leur perceptions).

Une fois bien maîtrisé ces notions, il est maintenant possible d'adapter son propre problème à un environnement markovien pour permettre l'apprentissage par renforcement. Pour les systèmes réels complexes, l'œil de l'expert permettra de choisir les bonnes approximations. Une activité intéressante est alors le passage de l'inférence du réseau de neurones obtenue en simulation sur le système réel. Des méthodes nommées Sim2Real existent pour rendre plus robuste ce passage. Un exemple de voitures autonomes est accessible sur le site culture Science de l'ingénieur :

https://eduscol.education.fr/sti/si-ens-paris-saclay/actualites/course-de-voitures-autonomes-2022-resultats

Références bibliographiques

[1] https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules

[2] R. S. Sutton, A. G. Barto, Reinforcement Learning An Introduction : Second Edition, MIT Press, 2014
https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[2] La documentation de Stablebaselines
https://stable-baselines.readthedocs.io/en/master/

[3] L. Da Costa, T. Parr, N. Sajid, S. Veselic, V. Neacsu, K. Friston, « Active inference on discrete state-spaces: A synthesis », Journal of Mathematical Physchology, 99, 2020.

[4] La documentation de l'environnement Gym, https://www.gymlibrary.ml/

[5] https://www.mlq.ai/deep-reinforcement-learning-q-learning/

[6] https://www.tensorflow.org/agents/tutorials/0_intro_rl

[7] https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

[8] https://ichi.pro/fr/decisions-a-partir-des-donnees-comment-l-apprentissage-par-renforcement-hors-ligne-changera-la-facon-dont-nous-utilison-40516786874800

[9] https://medium.datadriveninvestor.com/which-reinforcement-learning-rl-algorithm-to-use-where-when-and-in-what-scenario-e3e7617fb0b1

[10] QrCode amenant vers le dépôt des fichiers source :
https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/introduction-a-lapprentissage-par-renforcement

SERIES TEMPORELLES ET RESEAUX DE NEURONES RECURRENTS

VALENTIN NOËL
Doctorant SATIE, ENS Paris-Saclay & L2S, CentraleSupélec
Valentin.noel1@ens-paris-saclay.fr

Résumé : *En opposition aux données dites « statiques » (i.e images), les données dites « dynamiques » (i.e vidéos), dont font partie les séries temporelles, sont utilisées dans de nombreux domaines, allant de la météorologie à la finance, en passant par l'analyse de sentiments. Du fait de leur temporalité, les séries temporelles ne peuvent pas être fournies au réseau de neurones tel que le sont les données statiques. De plus, de nouvelles problématiques émergent dues elles aussi à leur temporalité, ce qui rend l'approche théorique et pratique de la prédiction de séries temporelles par réseaux de neurones récurrents (RNN) drastiquement différente de celle abordée pour des données statiques. Dans cet article, nous nous efforçons de dépeindre au mieux les caractéristiques, particularités et les complexités d'un tel type de données, que nous illustrons avec une application [7] : l'entraînement d'un RNN (i.e Long-Short Term Memory) pour la prévision de conditions climatiques.*

I/ Introduction

Dans le but de donner une image globale des différentes étapes de la prévision de données dynamiques il est nécessaire de définir les données dont nous voulons prévoir l'évolution. Nous prenons l'exemple de la consommation électrique d'un foyer, ce foyer désirant pouvoir prévoir l'évolution de cette consommation électrique d'une année sur l'autre afin de définir un budget prévisionnel. Ce phénomène n'étant pas parfaitement périodique car sujet à des aléas multiples : conditions météorologiques, nombre d'appareils électroniques, etc. Pour pouvoir pour autant prévoir l'évolution de ces données dynamiques, nous utilisons un réseau de neurones récurrent. Les données d'entraînement seront donc les consommations électriques des années précédentes, tandis que les données de sortie du réseau de neurones seront les données de la consommation électrique de l'année suivante.

II/ Séries temporelles : définition et spécificités

II.1/ *Des données statiques aux données dynamiques.*

Les séries temporelles sont un type de données qui sont échantillonnées en fonction d'une dimension temporelle (jours, mois, années, etc). Nous qualifions ces données de "dynamiques" car elles sont indexées sur la base d'un attribut de temps, ce qui leur donne donc un ordre temporel implicite. Les données statiques peuvent aussi avoir un attribut de temps, mais la différence réside dans le fait que les données statiques ne seront pas échantillonnées ou indexées en fonction de cet attribut.

Lorsque nous appliquons des algorithmes d'apprentissage automatique sur des données de séries temporelles et que nous voulons faire des prédictions pour des futures valeurs temporelles, par exemple, prédire le nombre de nouveaux étudiants inscrits dans un établissement à partir des données des 10 années précédentes, ou prédire le temps pour un certain jour à partir des données météorologiques de plusieurs années. Ces prédictions sur des données de séries temporelles sont appelées prévisions, ou « forecasting ».

II.2/ *Spécifités et contraintes des données temporelles*

Une contrainte inhérente à l'apprentissage par réseaux de neurones récurrents est de premièrement assurer la qualité des données dynamiques fournies, c'est-à-dire la continuité des séries temporelles fournies.

Ainsi, l'imputation des données manquantes est une étape clé du prétraitement dans tout projet d'apprentissage automatique tabulaire. Pour les données statiques, on peut utiliser des techniques telles que l'imputation simple, qui permet de combler les données manquantes par la moyenne, la médiane ou le mode des données, selon la nature de l'attribut, ou des méthodes plus sophistiquées telles que l'imputation par le plus proche voisin, qui utilise un algorithme KNN (K Nearest Neighbors) pour identifier les données manquantes.

Dans le cas des données dynamiques, la moyenne statique n'est d'aucune utilité, car il n'est pas forcément logique de combler les valeurs manquantes en prenant des repères dans le futur (en fonction des données disponibles). Nous utilisons ce que l'on appelle la moyenne glissante, la moyenne mobile ou la moyenne de fenêtre, qui consiste à prendre la moyenne des valeurs relatives à une fenêtre prédéfinie, par exemple une fenêtre de 7 jours ou d'un mois. Nous pouvons donc utiliser cette moyenne mobile pour combler les lacunes de nos séries temporelles.

Une méthode plus plébiscitée existe, qui utilise l'ordre implicite des données de séries chronologiques. L'interpolation est la méthode la plus utilisée pour déterminer les parties manquantes des données de séries temporelles. Les interpolations utilisent la valeur présente avant et après le point manquant pour calculer la donnée manquante. Par exemple, les interpolations linéaires fonctionnent en calculant une ligne droite entre

les deux points, en en faisant la moyenne et en obtenant la donnée manquante. Il existe de nombreux types d'interpolations, comme les interpolations linéaires, de Spline et de Stineman. Dans certains cas où de nombreuses données sont manquantes, d'autres approches basées sur des modèles auto-régressifs (AR), peuvent être utilisés. Les figures Fig.1 et Fig.2 représentent le signal d'entrée avec des données manquantes puis avec le signal reconstruit.

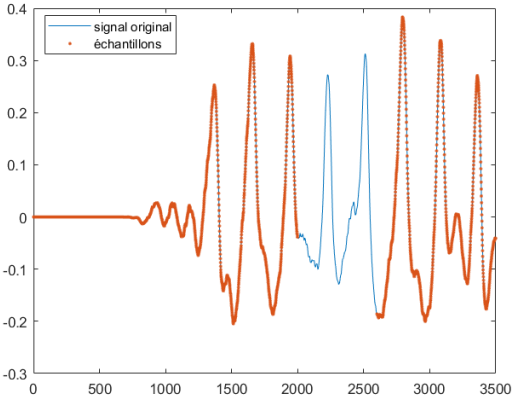


Figure 1 : Représentation du signal original avec le signal possédant des données manquantes

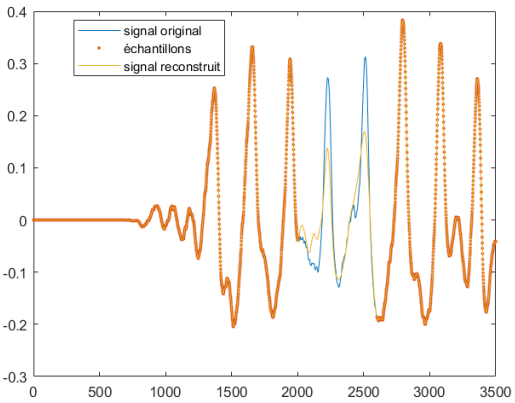


Figure 2 : Représentation du signal original avec le signal possédant des données manquantes et le signal reconstruit

II.3/ *Caractéristiques principales des séries temporelles*

Manipuler des caractéristiques différencie de nouveau les données dynamiques des données statiques : en effet, les caractéristiques sont traitées différemment dans les données de séries temporelles par rapport aux données statiques.

Dans le cas des données statiques, les techniques les plus communément rencontrées comprennent les transformations de caractéristiques, la mise à l'échelle, la compression, la normalisation, etc.

Quant aux données de séries temporelles, elles ont obligatoirement des composantes de séries temporelles. L'extraction de ces caractéristiques peut s'effectuer via la décomposition STL (Seasonal and Trend decomposition using Loess) et certaines d'entre elles sont définies ci-dessous [1]:

- **Saisonnalité** : La saisonnalité fait référence à une propriété des séries temporelles qui affiche des modèles périodiques se répétant à une fréquence constante.
- **Tendance** : Les données de séries chronologiques présentent une tendance lorsque leur valeur varie dans le temps. Une valeur croissante indique une tendance positive et une valeur décroissante, une tendance négative.
- **Reste** : Après avoir extrait la tendance et la saisonnalité des données, ce qui reste est ce que nous appelons le reste (erreur) ou le résidu. Cela permet de détecter les anomalies dans les séries chronologiques.
- **Cycle** : Les données de séries temporelles sont dites cycliques lorsqu'il existe des tendances sans répétitions fixes ou saisonnalité.
- **Stationnarité** : Les données de séries temporelles sont stationnaires lorsque leurs caractéristiques statistiques ne changent pas dans le temps, c'est-à-dire une moyenne et un écart-type constants.

Une fois extraites, ces composantes constituent la base pour l'analyse de l'évolution de la série temporelle afin de comprendre son comportement et de pouvoir choisir et adapter un modèle de série temporelle approprié.

Les données de séries temporelles peuvent avoir d'autres attributs que les caractéristiques temporelles. Si ces attributs sont temporels, la série temporelle résultante sera multivariée et si elle est statique, elle sera univariée avec des caractéristiques statiques. Les caractéristiques non temporelles peuvent utiliser des méthodes issues des techniques statiques de manière à ne pas nuire à l'intégrité des données.

III/ Approches algorithmiques pour la prévision des séries temporelles

III.1/ *Préparation des données dynamiques*

Il y a deux choses à établir avant de concevoir un modèle de prévision [2]:

- Les informations disponibles au moment où la prévision est faite (caractéristiques)
- La période pendant laquelle l'utilisateur a besoin de valeurs prévisionnelles (objectif).

L'origine de la prévision est le moment auquel une prévision est effectuée. En pratique, l'origine de la prévision est le dernier moment pour lequel des données d'entraînement sont disponibles pour la période à prévoir. Tout ce qui va jusqu'à l'origine peut être utilisé pour créer des caractéristiques.

L'horizon de prévision, qui décrit l'objectif, est la période pour laquelle une prévision va être effectuée. Une prévision est décrite la plupart du temps par le nombre de pas de temps dans son horizon : une prévision "à 1 pas" ou "à 5 pas", par exemple. Ce temps entre

l'origine et l'horizon est le délai d'exécution (parfois appelé la latence) de la prévision. Dans la pratique, il peut être nécessaire qu'une prévision commence plusieurs étapes avant l'origine en raison de retards dans l'acquisition ou le traitement des données.

III.2/ *Les réseaux de neurones récurrents*

Le choix des algorithmes pour les données de séries temporelles est complètement différent de celui des données statiques. Un algorithme applicable aux données temporelles doit être capable d'extrapoler des modèles et d'encapsuler les composantes de séries temporelles en dehors du domaine des données d'apprentissage. Or, la plupart des algorithmes d'apprentissage automatique statiques, comme la régression linéaire ou les SVM (Support Vector Machine), n'ont pas cette capacité car ils généralisent l'espace d'apprentissage pour toute nouvelle prédiction. Dans le but de prévoir les séries temporelles, des réseaux de neurones sont entre autres utilisés. Ces réseaux de neurones font partie de la famille des RNN (Recurent Neural Network). Le terme RNN est utilisé pour désigner la classe des réseaux à réponse impulsionnelle infinie, tandis que le terme CNN (Convolutional Neural Network) désigne une classe des réseaux à réponse impulsionnelle finie. Les deux classes de réseaux présentent un comportement dynamique temporel. Un réseau récurrent à réponse impulsionnelle finie est un graphe acyclique dirigé qui peut être déroulé dans le temps (foldable) et remplacé par un réseau neuronal strictement feedforward, tandis qu'un réseau récurrent à réponse impulsionnelle infinie est un graphe cyclique dirigé qui ne peut pas être déroulé dans le temps (unfoldable). Pour n'en citer que les principaux :

- FRNN (Fully Recurrent Neural Network)
- ESN (Echo State Network)
- Second-order RNNs:
  - LSTM (Long-Short Term Memory)
  - GRU (Gated Recurrent Unit)

III.3/ *Fully Recurrent Neural Network et le problème de gradient évanescent*

Dans le but de décrire de la façon la plus complète le fonctionnement d'un réseau de neurones récurrent, il est nécessaire de comprendre le comportement du FRNN et de la raison de l'apparition des RNNs de second ordre. Les FRNN connectent les sorties de tous les neurones aux entrées de tous les neurones. Il s'agit de la topologie de réseau neuronal la plus générale. La figure Fig.3 représente les différentes étapes dans le temps du même réseau neuronal entièrement récurrent.

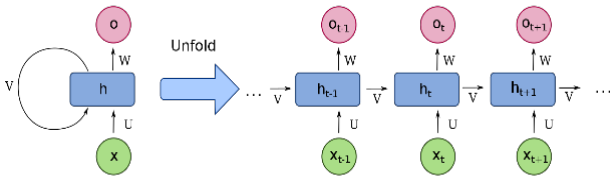


Figure 3 : De gauche à droite : RNN basique non-déplié dans le temps et RNN basique déplié dans le temps [3]

L'élément le plus à gauche de la figure Fig.3 illustre les connexions récurrentes sous la forme de l'arc étiqueté "v". Il est "déplié" dans le temps pour produire l'apparence de couches.

Cependant, un tel réseau est victime de ce qui est appelé dans la littérature le gradient évanescent (ou vanishing gradient). Le problème du gradient évanescent est rencontré lors de l'apprentissage des réseaux neuronaux artificiels avec des méthodes d'apprentissage basées sur le gradient et la rétropropagation. Dans ces méthodes, à chaque itération de la formation, chaque poids w_{ij} du réseau neuronal reçoit une mise à jour proportionnelle à la dérivée partielle de la fonction d'erreur par rapport au poids actuel. Les fonctions d'activation traditionnelles, telles que la fonction tangente hyperbolique, ont des gradients compris dans la plage (0,1). La rétropropagation calculant les gradients par la règle de la chaîne, avec n le nombre de couches du réseau, cela a pour effet de multiplier n de ces faibles valeurs pour calculer les gradients des premières couches du réseau. Cela signifie que le gradient diminue exponentiellement avec n et donc les premières couches s'entraînent très lentement. Dans certains cas, le gradient est extrêmement faible, ce qui empêche le poids de changer de valeur et ainsi mène à un apprentissage incomplet.

Le problème du gradient évanescent affecte non seulement les réseaux feedforward à plusieurs couches, mais aussi les réseaux récurrents. Lorsqu'on utilise des fonctions d'activation dont les dérivées peuvent prendre des valeurs plus importantes, on risque de rencontrer le problème connexe du gradient explosif.

III.4/ *Long-Short Term Memory, Gated Reccurent Unit: éviter le problème de gradient évanescent*

En théorie, les RNN classiques peuvent suivre des dépendances à long terme arbitraires dans les données d'entrée. Le problème des RNN classiques est de nature pratique : lors de l'apprentissage d'un RNN classique par rétropropagation, les gradients à long terme qui sont rétropropagés peuvent « disparaître » (c'est-à-dire tendre vers zéro) ou « exploser » (c'est-à-dire tendre vers l'infini), en raison des calculs impliqués dans le processus, qui utilisent des nombres à précision finie. Dans le but d'éviter ces problèmes de gradient, un réseau neuronal appelé Long-Short Term Memory (LSTM) a été développé.

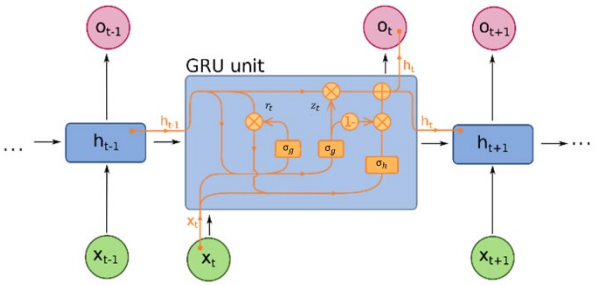


Figure 4: Long-Short Term Memory Unit [4]

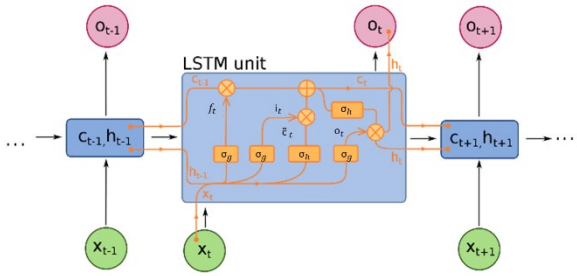


Figure 5: Gated Reccurent Unit [5]

Contrairement aux réseaux neuronaux feedforward standard, LSTM possède des connexions de rétroaction. Un tel réseau neuronal récurrent peut traiter non seulement des points de données statiques, mais aussi des données dynamiques. Une unité LSTM commune est composée d'une cellule, d'une porte d'entrée, d'une porte de sortie et d'une porte d'oubli, comme illustré dans la figure Fig.4. La cellule se souvient des valeurs sur des intervalles de temps arbitraires et les trois portes régulent le flux d'informations entrant et sortant de la cellule, ce qui a pour effet de limiter le problème du gradient évanescent. Les formes compactes des équations pour le forward pass d'une cellule LSTM avec une porte d'oubli sont [4] :

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_g(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \sigma_h(c_t)$$

Avec les valeurs initiales $c_0 = 0$ et $h_0 = 0$. L'opérateur $\circ$ dénote le produit d'Hadamard (produit par éléments). L'indice t représente le pas temporel. Les autres variables sont listées ci-dessous :

- $x_t \in R^d$: vecteur d'entrée
- $f_t \in (0,1)^h$: vecteur d'activation de la porte d'oubli
- $i_t \in (0,1)^h$: vecteur d'activation de la porte d'entrée/mise à jour
- $o_t \in (0,1)^h$: vecteur d'activation de la porte de sortie
- $h_t \in (-1,1)^h$: vecteur de sortie de l'unité LSTM.
- $\tilde{c}_t \in (-1,1)^h$: vecteur d'activation de l'entrée de la cellule
- $c_t \in R^h$: vecteur d'état des cellules
- $W \in R^{h \times d}, U \in R^{h \times h}$ et $b \in R^h$: les matrices de poids et les paramètres des vecteurs de biais qui doivent être appris pendant la formation

Les RNN utilisant des unités LSTM résolvent donc partiellement le problème du gradient évanescent. Cependant, les réseaux LSTM peuvent toujours souffrir du problème du gradient explosif. Quant au GRU, il est comme un LSTM avec une porte d'oubli, mais a moins de paramètres que le LSTM, car il manque une porte de sortie comme illustré dans la figure Fig.5.

Malgré cela, il a été démontré que les GRU ont une meilleure performance sur certains ensembles de données plus petits et moins fréquents.

IV/ *Application : séries temporelles de données climatiques.*

IV.1/ *Séries temporelles de données climatiques : le dataset.*

Bien qu'il en existe de plus en plus, les bibliothèques de référence pour construire des applications d'apprentissage automatique basées sur LSTM sont Tensorflow et Keras qui jouissent d'une forte popularité parmi les praticiens de l'apprentissage automatique. D'ailleurs, l'application montée ici est reproductible sur Google Collaboratory [7] : la prévision de séries temporelles pour les prévisions météorologiques.

Index	Features	Format	Description
1	Date Time	01.01.2009 00:10:00	Date-time reference
2	p (mbar)	996.52	The pascal SI derived unit of pressure used to quantify internal pressure. Meteorological reports typically state atmospheric pressure in millibars.
3	T (degC)	-8.02	Temperature in Celsius
4	Tpot (K)	265.4	Temperature in Kelvin
5	Tdew (degC)	-8.9	Temperature in Celsius relative to humidity. Dew Point is a measure of the absolute amount of water in the air, the DP is the temperature at which the air cannot hold all the moisture in it and water condenses.
6	rh (%)	93.3	Relative Humidity is a measure of how saturated the air is with water vapor, the %RH determines the amount of water contained within collection objects.
7	VPmax (mbar)	3.33	Saturation vapor pressure
8	VPact (mbar)	3.11	Vapor pressure
9	VPdef (mbar)	0.22	Vapor pressure deficit
10	sh (g/kg)	1.94	Specific humidity
11	H2OC (mmol/mol)	3.12	Water vapor concentration
12	rho (g/m ** 3)	1307.75	Airtight
13	wv (m/s)	1.03	Wind speed
14	max. wv (m/s)	1.75	Maximum wind speed
15	wd (deg)	152.3	Wind direction in degrees

Table 1 : Détail des caractéristiques du set de données

Nous utiliserons le jeu de données climatiques d'Iéna (Allemagne) enregistré par l'Institut Max Planck de biogéochimie. Le jeu de données se compose de 14 caractéristiques telles que la température, la pression, l'humidité, etc., enregistrées une fois toutes les 10 minutes. Le tableau Table 1 présente les noms des colonnes, leurs formats de valeur et leur description.

IV.2/ *Visualisation des données brutes*

Pour nous donner une idée des données avec lesquelles nous travaillons, chaque caractéristique a été tracée ci-dessous dans la figure Fig.6.

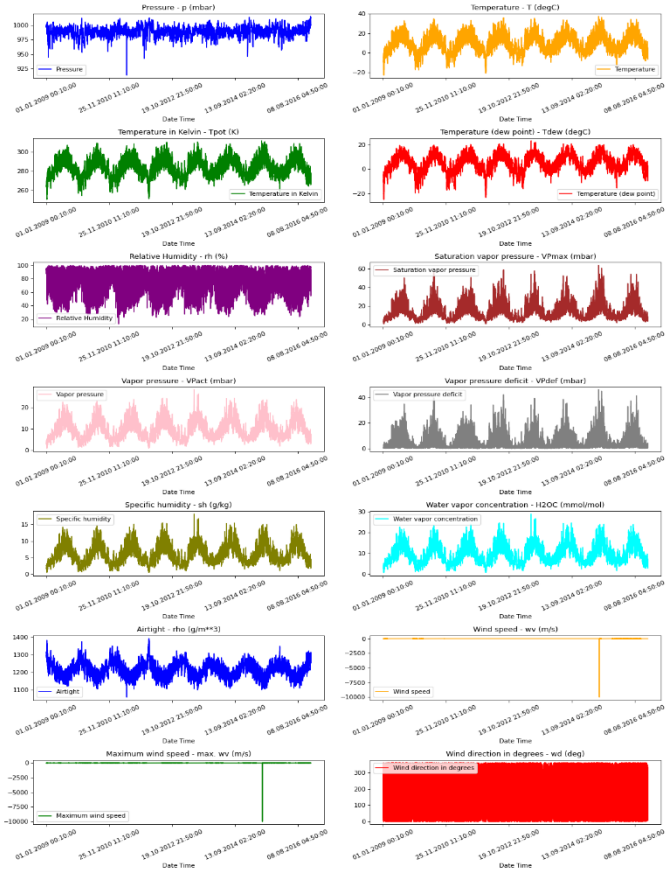


Figure 6 : Visualisation des données brutes du set de données

Cela montre le modèle distinct de chaque caractéristique sur la période de 2009 à 2016. Il montre également où se trouvent les anomalies, qui seront traitées lors de l'étape de normalisation. De façon à visualiser la corrélation entre différentes caractéristiques, une heat map est disponible à la figue Fig.7. Nous pouvons voir sur la heat map de corrélation que certains paramètres comme l'humidité relative et l'humidité spécifique sont redondants. Par conséquent, nous utiliserons certaines caractéristiques, pas toutes, certaines seront enlevées lors de l'étape de pré-traitement des données.

IV.3/ *Prétraitement des données*

Ici, ~300 000 points de données sont choisis pour l'entraînement. L'observation est enregistrée toutes les 10 minutes, c'est-à-dire 6 fois par heure. Le rééchantillonnage est mis à un point par heure car aucun changement radical n'est attendu en 60 minutes. Les 1440 derniers horodatages (1440/6=240 heures) sont suivis. Ces données seront utilisées pour prédire la température après 72 horodatages (72/6=12 heures). Comme chaque caractéristique a des valeurs avec des plages variables, une normalisation est effectuée pour confiner les valeurs des caractéristiques dans une plage

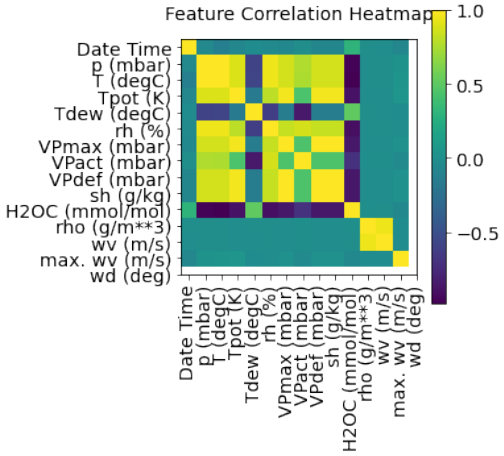


Figure 7 : Heat map des corrélations entre caractéristiques du set de données

de [0, 1] avant de former un réseau neuronal. Pour ce faire, la moyenne est soustraite à chaque valeur et est divisée par l'écart type de chaque caractéristique. 70 % des données seront utilisées pour entraîner le modèle, c'est-à-dire 294 385 lignes. Le modèle reçoit les données des 10 premiers jours, soit 1440 observations, qui sont échantillonnées toutes les heures. La température après 72 (12 heures * 6 observations par heure) observations sera utilisée comme étiquette.

IV.4/ *Les sets d'entraînement et de validation*

Pour le set d'entraînement, les labels de l'ensemble de données d'apprentissage commencent à partir de la $1512^{ème}$ observation (1440 + 72). Une fonction est définie, ayant pour rôle de prendre une séquence de points de données collectés à intervalles égaux, ainsi que des paramètres de séries temporelles tels que la longueur des séquences/fenêtres, l'espacement entre deux séquences/fenêtres, etc., pour produire des lots de sous-séries d'entrées et de cibles échantillonnées à partir de la série temporelle principale.

L'ensemble de données de validation ne doit pas contenir les 1512 dernières lignes, car nous n'aurons pas de données d'étiquette pour ces enregistrements, donc 1512 doit être soustrait de la fin des données. L'ensemble de données de validation doit commencer à partir de 1512 après la dernière donnée utilisée pour le set de test.

Il doit être noté que les paramètres choisis ici sont différents de ceux proposés dans [7], pouvant mener à des résultats non-optimaux. Ce choix a été fait dans le but de pouvoir comparer les résultats obtenus dans les deux configurations mais aussi d'illustrer l'intérêt de l'utilisation de notebook facilement modifiables et très visuels pour la compréhension de thèmes plus ou moins compliqués, comme ici les réseaux de neurones récurrents.

IV.5/ *Entraînement et prédiction*

Le ModelCheckpoint Callback est utilisé pour sauvegarder régulièrement les points de contrôle, et l'EarlyStopping callback pour interrompre la formation lorsque la validation loss ne s'améliore plus. L'évolution de la validation loss et training loss est représentée dans la figure Fig.8.

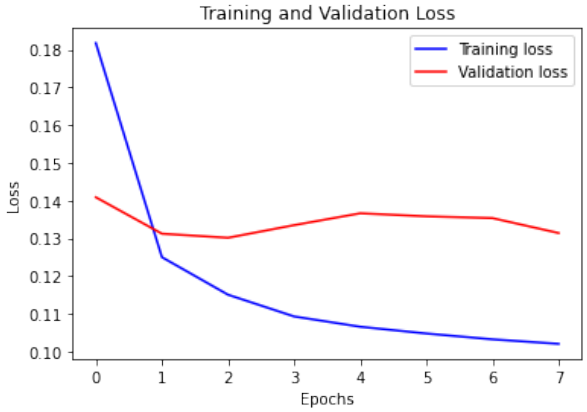


Figure 8 : Evolution de la validation loss et training loss en fonction du nombre d'epochs.

Nous remarquons que ces paramètres mènent à de meilleurs metrics finaux, comparativement à ceux utilisés en [7].

Le modèle ainsi entraîné est maintenant capable d'effectuer des prédictions pour 5 sets de données issues du set de validation. Nous décidons ici de mettre en avant la prédiction obtenue pour 1 de ces 5 sets dans la figure Fig.9.

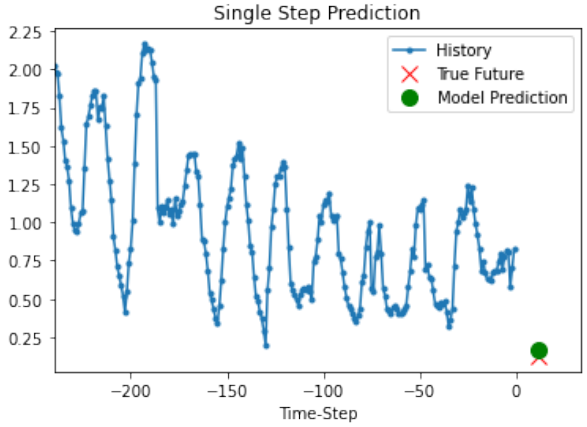


Figure 9 : Prédiction obtenue par le modèle entraîné.

Nous observons que la prédiction pour ce set de données est très bonne. D'autres tests peuvent être menés afin de déterminer les paramètres menant aux meilleurs résultats.

V/ Conclusion

Dans cet article ont été mis en avant les fondements des réseaux de neurones récurrents et des séries temporelles. Les principales problématiques liées aux données dynamiques ont donc été abordées, ainsi que leurs principales caractéristiques et donc leur différence avec les données statiques. En suivant le même raisonnement, il a été expliqué pourquoi la majorité des algorithmes applicables aux données statiques ne pouvaient être appliqués aux séries temporelles. Suivant cette explication, différentes architectures de réseaux ont été proposées avec l'objectif d'illustrer clairement le fonctionnement général dans un premier temps, puis spécifique de second ordre.

Cependant, nous ne pouvons que mettre en avant le nombre limité de réseaux neuronaux montrées ici. Il est de la même façon nécessaire de citer d'autres approches tout aussi viables, qui n'ont pas été abordées afin de ne pas augmenter inutilement la complexité de l'article : Autoregressive-Integrated-Moving-Average (ARIMA), Prophet et Exponential Smoothening (EWMA) sont d'autres approches que nous recommandons d'étudier en parallèle afin d'avoir un aperçu d'autant plus global de la prévision de séries temporelles [6].

VI/ Bibliographie

[1] https://neptune.ai/blog/time-series-prediction-vs-machine-learning

[2] https://www.kaggle.com/code/ryanholbrook/forecasting-with-machine-learning

[3] https://en.wikipedia.org/wiki/Recurrent_neural network

[4] https://en.wikipedia.org/wiki/Long_short-term_memory

[5] https://en.wikipedia.org/wiki/Gated_recurrent_unit

[6] Mahmoud, E., & Pegels, C. C. (1990). An approach for selecting times series forecasting models. *International Journal of Operations & Production Management*.

[7] https://keras.io/examples/timeseries/timeseries_weather_forecasting/

INTRODUCTION AUX BIBLIOTHEQUES GYM ET STABLE-BASELINES POUR L'APPRENTISSAGE PAR RENFORCEMENT

GUENOLE CHEROT [1], AUGUSTIN GODINOT [2]
[1] : Doctorant au laboratoire SATIE de l'ENS Rennes, guenole.cherot@ens-rennes.fr
[2] : Elève au DER Nikola Tesla de l'ENS Paris Saclay, augustin.godinot@ens-paris-saclay.fr

Résumé *: Cet article présente les très populaires bibliothèques Gym et Stable-Baselines dédiées à l'apprentissage par renforcement. Il s'appuie sur une séance de travaux pratiques d'asservissement d'un pendule inversé.*

Cette séance de travaux pratiques est aussi l'occasion de présenter le pédagogique outil jupyter notebook.

Mots clés *: apprentissage par renforcement profond, reinforcement learning, pendule inversé, Cartpole, Gym, Stable-Baselines, Jupyter notebook*

I/ Faire avancer la science reproductible

Malgré la simplicité apparente de la formulation d'un problème d'apprentissage par renforcement (Processus de Décision Markovien, stationnaire), l'analyse théorique de ses performances sur un système particulier reste compliquée, voire impossible. La validation des algorithmes repose donc majoritairement sur l'évaluation de leurs performances expérimentales. Or, une validation expérimentale rigoureuse nécessite une attention particulière. L'expérience doit-être :

- Contrôlée : Le problème à résoudre doit être clairement défini.
- Répétable : Les résultats obtenus doivent être réplicables par d'autre scientifique de manière indépendante.
- Statistiquement significative : Les algorithmes étant de nature stochastique, un grand nombre d'expérience identique doit être mené pour limiter les erreurs d'interprétation.

De nombreux efforts sont déployés dans la communauté Machine Learning pour faire face à la crise de la reproductibilité [1]. Par exemple, la société OpenAI a créé en 2016 **Gym**[1] [2]. Cette librairie open-source implémente une grande diversité d'environnements (pendule inversé, jeux Atari, robots...) avec une interface unifiée. Pour faciliter encore d'avantage la comparaison d'algorithme, **Stable-Baselines**[2] [4] fourni une implémentation open-source de l'ensemble des algorithmes à l'état de l'art.

Prenons maintenant un exemple pratique de l'utilisation de ces deux outils : si un chercheur propose un nouvel algorithme, il va commencer par le tester sur l'ensemble des environnements **Gym**. Ces expériences, lui donneront des scores (somme des récompenses obtenu sur un épisode) sur chacun des environnements. Il pourra ensuite comparer ces scores avec ceux obtenus par les algorithmes de l'état de l'art grâce à la librairie **Stable-Baselines**. Cette procédure est illustrée Figure 1.

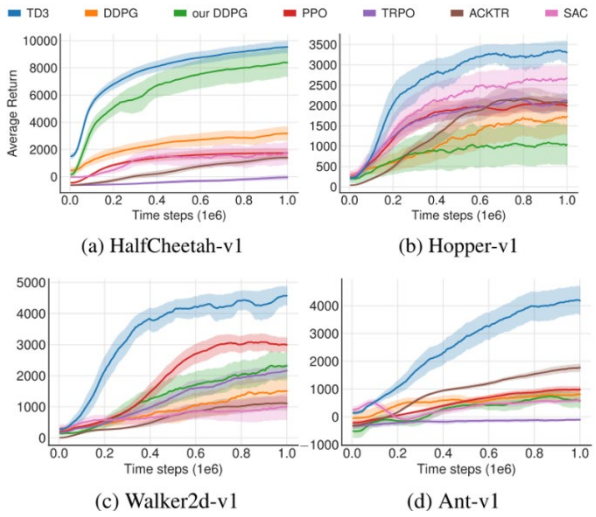


Figure 1 : Comparaison de sept algorithmes d'apprentissage par renforcement sur quatre environnements Gym [3]

Cet article présentera les librairies Gym et StableBaselines à travers une séance de travaux pratiques d'asservissement d'un pendule inversé. Aucune connaissance préalable de ces librairies n'est nécessaire, cependant, il est préférable d'avoir lu l'article *Introduction à l'apprentissage par renforcement* de ce numéro, introduisant les principes généraux de l'apprentissage par renforcement. Une bonne maitrise du langage de programmation Python est également conseillée.

[1] https://www.gymlibrary.ml/

[2] https://stable-baselines3.readthedocs.io/en/master/

II/ Gym et Stable-Baselines pour l'enseignement de l'apprentissage par renforcement.

La puissance de Gym et Stable-Baselines réside dans leur simplicité d'utilisation, plus précisément, la simplicité de leur interface. Dans une situation classique d'apprentissage par renforcement, à chaque instant t, un agent effectue une observation $o_t \in O$ de l'état $x_t \in X$ de son environnement. Suivant la valeur de son observation, il choisit d'effectuer une action $a_t \in A$. L'environnement récompense alors l'agent d'une valeur $r_t \in R$ dépendant de son action et de son état interne x_t.

II.1/ *Introduction sur un exemple*

Regardons, pas à pas, comment simuler cette situation avec Gym et comment y entraîner un agent avec Stable-Baselines. On s'intéresse au pendule inversé Cartpole, l'un des environnements les plus simples proposés par Gym.

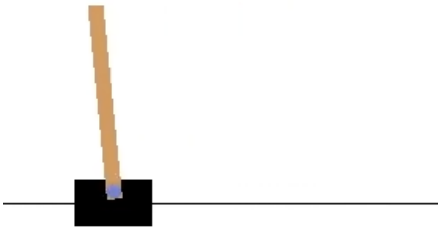


Figure 2 : Environnement Cartpole de la bibliothèque Gym

On choisit l'agent PPO (proximal policy optimization) de Stable-Baselines, l'un des plus populaires et génériques. Le code complet est extrêmement court, les bibliothèques étant de très haut niveau.

Les variables (o_t, a_t, r_t) de la formulation mathématique du problème sont facilement identifiable dans le code ((o_t, a_t, r_t) ≡ (obs, action, reward)), tout comme leur dépendance. En pratique, un agent n'a jamais une connaissance parfaite de son environnement, l'état x_t de celui-ci n'apparaît donc pas parmi les variables manipulables dans le code.

Chaque algorithme de Stable-Baseline est une classe Python qui implémente une méthode `predict(observation)` prenant une observation en entrée et retournant une action, c'est donc un accès direct à la politique π de l'agent. En plus de proposer un ensemble d'agents (i.e. algorithmes retournant une action pour une observation) implémentés, Stable-Baseline se charge aussi de les entraîner via la fonction `model.learn()` de la ligne 17.

[3] https://colab.research.google.com/drive/1AZh_FSSImncpOuy_vdGbewK6OyqksAme

```
9  from stable_baselines3 import PPO
10 import gym
11
12 # Création de l'environnement
13 env = gym.make("CartPole-v1")
14
15 #Lancement de l'apprentissage avec l'alorithme PPO
16 model = PPO(policy = "MlpPolicy",env =  env, verbose=1)
17 model.learn(total_timesteps=25000)
18
19 # (ré-)initialisation de l'environnement
20 obs = env.reset()
21
22 # Simule le rejeu 1000 pas de temps dans l'environnement
23 for i in range(1000):
24     # Selection de l'action à effectuer
25     action, _state = model.predict(obs)
26     # Appliquer l'action
27     obs, reward, done, info = env.step(action)
28     # Afficher
29     env.render()
30     # Si la simulation est terminé,
31     # on remet le pendule au centre
32     if done:
33       obs = env.reset()
```

Figure 3 : Code complet de l'apprentissage de l'asservissement d'un pendule inversé avec Gym et Stable-Baselines

II.2/ *TP sur JupyterNotebook*

L'exercice proposé utilise aussi le pendule inversé nommé Cartpole pour mettre en œuvre un apprentissage par renforcement avec la bibliothèque Stable-Baselines.

Il utilise un jupyter notebook hébergé par googlecolab[3] pour alterner code et explications :

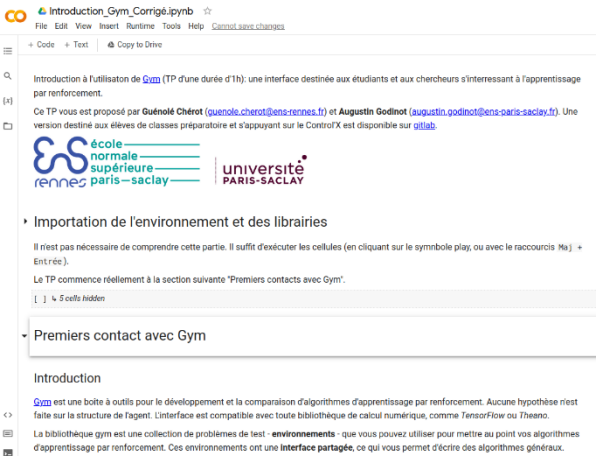


Figure 4 : Page d'accueil de la séance de travaux pratiques pendule inversé avec Gym et Stable-Baselines

La consultation du site montrera l'intérêt de l'outil jupyter Notebook pour la pédagogie. Les lignes de codes sont exécutées bloc par bloc, les explications accompagnent chaque bloc. Il est possible de demander aux étudiants de compléter certains blocs.

III/ Conclusion

Grâce aux bibliothèques Gym et Stable-Baselines, il est possible de simuler un environnement et un agent en une dizaine de lignes de Python. Les variables manipulées par les implémentations reflètent la modélisation mathématique du problème, ce qui facilite la prise en main et l'apprentissage par les étudiants. En bref, Gym fournit un ensemble de problèmes à résoudre pour comparer les agents sur une même base. Stable-Baselines fournit un ensemble d'agents fonctionnels à utiliser en enseignement et auxquels se comparer lors de la conception de nouveaux agents.

Il est également possible d'utiliser avec Gym et Stable-Baselines son propre environnement pour travailler sur un système original. L'article *Apprentissage par renforcement de la conduite d'un véhicule sur AirSim* présente une utilisation de Gym et Stable-Baselines pour de l'apprentissage par renforcement avec le simulateur de voitures autonomes réaliste AirSim.

IV/ Bibliographie

[1] Baker, M. (2016). Reproducibility crisis. *Nature*, *533*(26), 353-66.

[2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[3] Fujimoto, S., Hoof, H., & Meger, D. (2018, July). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587-1596). PMLR.

[4] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*

APPRENTISSAGE PAR RENFORCEMENT DE LA CONDUITE D'UN VEHICULE SUR AIRSIM

LUDOVIC DE MATTEIS [1], SASA RADOSAVLJEVIC [2]
[1] : élève du département Nikola Tesla de l'ENS Paris Saclay, ludovic.de_matteis@ens-paris-saclay.fr
[2] élève du département Nikola Tesla de l'ENS Paris Saclay, sasa.radosavljevic@ens-paris-saclay.fr

Résumé *: Cet article propose d'une part une application pédagogique concrète et attrayante de l'apprentissage par renforcement : l'apprentissage réaliste de la conduite autonome sur le simulateur AirSim. D'autre part, il présente l'intégration d'un environnement de simulation aux bibliothèques Python Gym et Stable Baselines, facilitant l'utilisation des algorithmes d'apprentissage par renforcement.*
Mots clés *: Voiture autonome, apprentissage par renforcement, AirSim*

I/ Introduction

L'émergence des transports autonomes, notamment les voitures autonomes, oblige le développement de nouvelles méthodes de conduite. Plusieurs méthodes d'intelligence artificielle peuvent être utilisées afin de créer des systèmes autonomes performants. Dans cet article, nous nous intéresserons particulièrement à l'utilisation de l'apprentissage par renforcement pour répondre à notre problème.

Cette étude étant particulièrement adaptée à une séance de travaux pratiques, l'article s'attache à détailler les différents éléments de l'apprentissage par renforcement et leur implémentation, en utilisant des outils logiciels adaptés. Les activités peuvent être effectuées à différents niveaux d'études en s'attachant plus ou moins à la construction de chacun des éléments (agent, environnement, récompenses, observations…). Les résultats des phases d'entrainements et la progression de l'agent présentent clairement l'apport de l'apprentissage par renforcement pour la résolution de problèmes complexes.

Cet article propose une méthode d'intégration d'un simulateur dans la démarche d'apprentissage par renforcement. Cette méthode peut être réutilisée et adaptée à d'autres problèmes d'apprentissage pour lesquels un simulateur est accessible. Cet article ne détaillera pas les concepts de l'apprentissage par renforcement mais s'orientera seulement vers leur mise en œuvre dans le cadre du problème de conduite autonome. Les aspects théoriques de l'apprentissage par renforcement peuvent être retrouvés dans d'autres articles du dossier IA de la revue 3EI.

II/ Déroulement de la séance de travaux pratiques

II.1/ *Objectif de la séance*

Cette séance d'activité pratique a pour objectif de présenter les concepts d'apprentissage par renforcement par un exemple complexe concret.

La problématique centrale de cette séance est celle de la conduite autonome. La complexité de la tâche est telle qu'il est difficile de la résoudre de manière performante avec des outils classiques de planification de trajectoire et d'évitement d'obstacles. Nous allons alors nous intéresser à la faisabilité de l'utilisation d'outils d'intelligence artificielle pour résoudre, même partiellement, ce problème. La disponibilité d'un simulateur rend le problème très adapté à une résolution par apprentissage par renforcement.

Cette séance commencera par présenter les éléments fondamentaux de l'apprentissage par renforcement, à savoir la définition de l'espace des observations, de l'espace des actions et de l'agent. Par la suite, l'implémentation de l'algorithme d'apprentissage sera présentée en utilisant les bibliothèques Python Gym et Stable Baselines. La séance est constituée de quelques questions ramenant aux concepts de bases d'apprentissage et d'étapes permettant de mettre en œuvre l'exemple étudié.

II.2/ *Support de l'activité*

Cette séance d'activité pratique se basera sur l'utilisation du simulateur AirSim[1]. Il s'agit d'un simulateur de drone et de voiture intégré au moteur physique Unreal Engine 4[2] (il existe également une intégration au moteur physique Unity, encore en développement). L'utilisation d'un simulateur pour cette activité permet de travailler sur un problème concret et réaliste même si l'accès ou l'utilisation du système réel est impossible. Dans le cadre de la conduite autonome l'utilisation d'un système réel n'est pas envisageable en raison des nombreuses collisions qui auront lieu lors de la première phase d'apprentissage. Enfin, en utilisant les fonctionnalités d'AirSim et d'Unreal Engine, il est possible d'accélérer le temps en simulation et donc d'apprendre plus rapidement. Ce dernier point est fondamental pour l'application des algorithmes d'apprentissage en séances de travaux pratiques, les phases d'apprentissage étant généralement longues.

[1]- https://microsoft.github.io/AirSim/

[2]- www.unrealengine.com/en-US

Dans le simulateur, la voiture par défaut est utilisée, à laquelle est ajouté un capteur LIDAR de paramètres réglables. Le paramétrage de la simulation est fait par un fichier *« settings.json »* se trouvant dans les documents utilisateur (l'installation du simulateur AirSim est décrite plus en détail dans la partie suivante).

II.3/ *Outils utilisés*

Question 1 :

Parmi les algorithmes de Reinforcement Learning que propose Stable Baselines, lequel semble le plus adapté selon vous (justifiez) ?

Réponse 1 :

L'algorithme Advantage Actor-Critic (A2C) peut être utilisé pour avec un espace des actions importants, cependant son utilisation favorise le multi-agent pour apprendre avec plusieurs environnements en parallèle, ce qui peut être contraignant niveau ressources. L'environnement simulé que nous utilisons demandant déjà beaucoup de ressources, cet algorithme ne semble pas adapté. L'agorithme Deep Q-Learning (DQN) demande un espace d'action discret mais propose d'utiliser un espace des observations important permettant la prise en compte des nombreuses mesures que peut procurer le Lidar. Cette méthode est bien adaptée à notre problème et nous utiliserons donc le DQN dans le projet présenté. L'algorithme Hindsight Experience Replay (HER) simplifie le système de récompense et utilise d'autres politiques d'apprentissage. Il peut être intéressant d'approfondir cette méthode afin d'observer ses performances.

- Description

Pour réaliser l'apprentissage par renforcement, il faut définir les éléments principaux. Ceux-ci sont représentés figure 1.

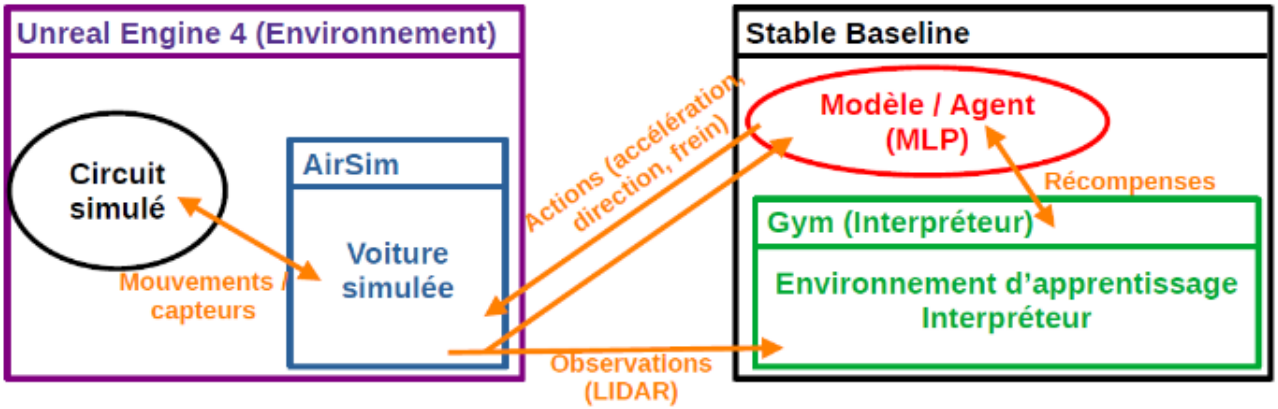


Figure 1 : Structure du problème et interaction des différents outils logiciels utilisés. Pour tout problème d'apprentissage par renforcement, si l'on dispose d'un environnement simulé, on peut appliquer la même démarche pour l'utiliser avec Gym et Stable Baselines

- L'agent, prenant des décisions en fonction des observations reçues est modélisé et géré par la bibliothèque Python Stable Baselines. C'est le conducteur autonome de la voiture. Il s'agit dans notre cas d'un réseau de neurones de type Multi Layer Perceptron (MLP) comportant 2 couches cachées de 64 neurones. La bibliothèque utilisée permet également de superviser la phase d'apprentissage en assurant le lien entre l'agent et l'interpréteur.
- L'interpréteur (ou environnement d'apprentissage), permet l'envoi d'observations de l'environnement simulé à l'agent, la définition des récompenses et leur envoi à l'agent et l'envoi d'actions de l'agent vers l'environnement simulé. Dans l'activité présentée ici, l'interpréteur sera réalisé à l'aide de la bibliothèque Python Gym.
- L'environnement simulé, comme discuté précédemment, permet de faire évoluer l'agent dans un circuit simulé en fonction des actions choisies et de lui retourner les informations des capteurs. Il comprend la voiture, ses capteurs et actionneurs, la piste, et leurs interactions. L'environnement est créé par le moteur physique Unreal Engine 4 et le simulateur AirSim y intègre la voiture.

- Installation des outils

Pour ce projet, Python est utilisé avec les bibliothèques Gym et Stable baselines. Stable baselines est une bibliothèque Python permettant l'implémentation de nombreux algorithmes d'apprentissage par renforcement. De nombreux paramètres sont réglables lors de l'utilisation de Stable baselines, notamment la politique de l'agent, l'algorithme d'apprentissage par renforcement et ses hyperparamètres, ainsi que l'environnement d'apprentissage. Cet environnement d'apprentissage est défini à l'aide de la bibliothèque Python Gym.

Le moteur physique Unreal Engine est utilisé avec l'extension AirSim. Pour ce projet (et pour des raisons de compatibilité) nous utilisons le système d'exploitation Windows 10 ainsi que Unreal Engine 4.27 et Visual Studio 2019. Afin de procéder à l'installation des logiciels utilisés et des dépendances Python, veuillez suivre les informations disponibles sur le dépôt Github suivant https://github.com/LudovicDeMatteis/Revue3EI_AirSim

Le dossier contient plusieurs éléments importants ; le dossier *PythonClient/dqn_car* contient les scripts Python utilisés lors de ce projet le dossier *Unreal/Environments* contient deux environnements différents pouvant être exécutés en lançant le projet nommé *Blocks.uproject* ou en lançant *Binaries/Win64/Blocks.exe* une fois l'environnement compilé par UE. Enfin, le fichier *settings.json* doit être déplacé dans un dossier nommé *AirSim* dans le dossier Documents de l'utilisateur Windows. Ce fichier contient plusieurs réglages de la simulation, notamment ceux du véhicule et des capteurs utilisés.

Plusieurs modèles pré-entrainés sont fournis et peuvent être directement utilisés, comme présentés dans les parties '*Démarrer l'apprentissage'* et *'Observer les résultats'* de cet article.

III/ Formalisme

III.1/ *Commandes du véhicule via AirSim*

Afin que l'agent (le véhicule autonome) puisse évoluer dans l'environnement simulé, il est nécessaire de définir les actions possibles. Dans le cadre d'un algorithme de Deep Q-Learning, que nous allons utiliser par la suite, l'espace des actions possible doit être discret. Nous allons donc chercher un espace des actions permettant de réaliser la tâche souhaitée.

Le simulateur AirSim permet de contrôler l'accélération du véhicule, la direction des roues et le freinage. L'accélération est commandée sous forme d'une grandeur scalaire comprise en 0 et 1, la direction est comprise entre -0,5 et 0,5 et le freinage est une grandeur binaire commandant l'appui ou non sur le frein.

Question 2 :

Proposez et justifiez un espace des actions permettant de répondre efficacement au problème (entraînement rapide, généralisable…) ?

Réponse 2 :

Il est important d'avoir suffisamment d'actions possibles pour que le véhicule puisse parcourir l'intégralité du parcours et puisse s'adapter à de nouveaux circuits. Ainsi, il semble intéressant d'utiliser l'amplitude totale accessible. Il faut cependant sélectionner un nombre fini d'actions et donc discrétiser l'espace continu proposé par AirSim.

Effectivement, un nombre trop faible d'actions ne laissera que peu de marge au véhicule pour s'adapter aux situations rencontrées. Par exemple, ne tourner qu'avec une valeur de braquage maximale à droite ou à gauche ne permet pas au véhicule d'adopter une conduite « optimale ». A l'inverse, un trop grand nombre de paramètres risque de ralentir l'apprentissage en obligeant le système à tester plus de combinaisons observations-actions.

Dans le cadre de l'activité pratique présentée ici, nous utiliserons 3 valeurs d'accélérations possibles (0, 0.5 et 1) et 5 valeurs de directions (-0.5, -0.25, 0, 0.25, 0.5). Ceci nous donne donc 15 actions possibles, représentées en figure 2, auxquelles s'ajoute une action de freinage sans braquage.

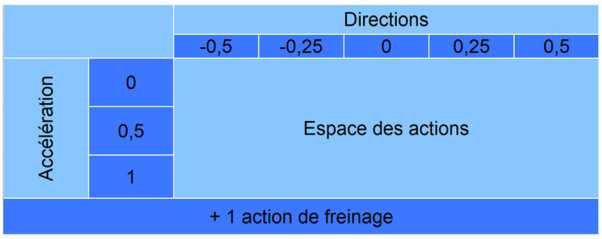


Figure 2 : Espace des actions utilisé pour la résolution de notre problème

La démarche présentée dans cette article reste valable pour un espace des actions différent. Il reste cependant important de garder une certaine symétrie dans la définition des directions afin d'éviter de biaiser ici la conduite vers la droite ou la gauche.

III.2/ *Retour d'informations de l'environnement*

Pour prendre une décision, l'agent a besoin d'avoir à disposition des informations sur son environnement. Ces informations, appelées observations, prennent la forme de données de capteurs. Pour le problème considéré, nous utiliserons un capteur LIDAR, comme mentionné précédemment. Il est cependant possible d'utiliser d'autres capteurs en changeant simplement le paramétrage du simulateur AirSim.

Le capteur que nous utilisons dans l'exemple présenté ici est paramétré avec 200 points par rotation et 10 rotations par secondes. Les faisceaux sont contenus dans un plan horizontal et avec un champ de vue de -120° à 120°. La figure 3 présente la voiture dans le simulateur AirSim ainsi que les points repérés par le capteur utilisé.

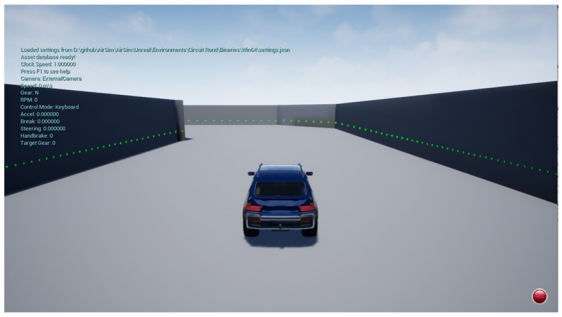


Figure 3 : Véhicule simulé et représentation des points détectés par le capteur LIDAR installé

Il est alors possible de récupérer directement dans un script Python les données du capteur. Une liste de 200 éléments est alors créée, contenant la position (x,y,z) de l'élément repéré par le LIDAR. La distance à l'obstacle ainsi que son angle associé sont obtenus à l'aide d'une conversion trigonométrique.

III.3/ *Récompenses*

La mise à jour de la politique de l'agent se fait par l'utilisation de récompenses. Une grande récompense indique que l'action choisie compte tenu des observations est bonne. A l'inverse, une récompense faible, voire négative, indique à l'agent que son choix n'était pas bon. L'apprentissage par renforcement optimise la

politique pour maximiser la récompense totale. Celle-ci a donc un rôle central dans l'apprentissage. Dans notre cas, la politique étant un réseau de neurones dense (MLP), la récompense reçue est utilisée sous la forme d'une fonction coût dans l'algorithme de rétropropagation.

Il est important de proposer des récompenses qui pousse l'agent vers le comportement désiré.

Question 3 :

Présentez quelques aspects du comportement souhaité du véhicule. Comment traduire ces comportements sous forme de récompenses ?

Réponse 3 :

Afin de progresser sur le circuit, il faut tout d'abord que le véhicule avance. Ce comportement peut être vu de plusieurs manières ; cela peut correspondre à un déplacement du véhicule entre deux instants ou à une vitesse non nulle du véhicule. Dans le premier cas, la position du véhicule peut être enregistrée à chaque instant et la distance euclidienne entre deux positions successives est choisie comme récompense R_d. Le second point de vue se traduit en utilisant simplement une fonction de la vitesseR_v.

Il faut également que la voiture reste éloignée des murs afin qu'elle progresse dans le circuit sans collisions. Une récompense possible pour favoriser ce comportement est la distance minimale à un obstacle. En effet, pour maximiser cette récompense, l'agent devra se trouver au milieu de la route. Cette récompense $R_p$est alors définie dans notre cas comme la valeur minimale contenue dans la liste des observations issues du LIDAR.

La récompense totale est alors définie comme $R = \alpha R_d + \beta R_p$ ou $R = \alpha R_v + \beta R_p$ avec α et β des paramètres réglables permettant d'ajuster les ordres de grandeurs dans différents éléments et d'accentuer l'un ou l'autre des comportements.

D'autres comportements peuvent être pensés et ajoutés sous forme de récompense, comme favoriser une trajectoire avec moins de virages (limitant l'oscillation du véhicule en ligne droite) ou mesurer l'avancée du véhicule par la traversée de points de contrôles.

IV/ Définir les éléments d'apprentissage

IV.1/ *Définition d'un environnement Gym*

Maintenant que les éléments principaux du problème sont définis (observations, simulation, récompenses), il est possible de mettre en place l'environnement d'apprentissage qui nous permettra d'entrainer l'agent. Pour cela, nous avons défini un environnement personnalisé en utilisant la bibliothèque Python Gym.

Pour permettre le fonctionnement de l'environnement d'apprentissage, il faut implémenter plusieurs méthodes particulières :

- *__init__* : cette méthode, constructeur de l'environnement d'apprentissage, **initialise l'agent** et les grandeurs qu'il utilisera, notamment l'espace des actions.

- *step* : cette méthode réalise une action de l'agent. Prenant en argument une action, elle **applique l'action**, c'est à dire la réalise en simulation, puis **récupère les observations** et **calcule les récompenses** qui seront transmises à l'algorithme d'apprentissage pour mettre à jour la politique

- *reset* : cette méthode **replace la voiture dans son état initial**. Afin de limiter les biais lors de l'apprentissage, la voiture a plusieurs points d'apparitions possibles. Le choix du lieu de réapparition est aléatoire.

A travers les différentes méthodes définies dans cet environnement, l'agent interagit avec l'environnement simulé. Cette interaction est possible grâce à l'utilisation de la bibliothèque fournie par AirSim, permettant la récupération des données des capteurs et l'envoi de commandes au véhicule.

IV.2/ *Définir l'agent*

Le dernier élément à définir pour permettre l'apprentissage est la politique de l'agent et l'algorithme d'apprentissage associé. Dans l'exemple fourni, et comme précisé précédemment, la politique de l'agent se base sur l'utilisation d'un réseau de neurones totalement connecté (MLP).

L'algorithme d'apprentissage utilisé est un algorithme de double deep Q-Learning (DQN).

Ces éléments sont définis à l'aide de la bibliothèque Python Stable Baselines et les hyperparamètres sont gardés à des valeurs par défaut pour la plupart. La définition du modèle et des valeurs des hyperparamètres ayant changé sont présentés en figure 4.

Il est intéressant d'observer les effets des différents hyperparamètres et changeant les paramètres avant l'apprentissage. En particulier, l'effet du taux d'apprentissage (*learning_rate*) et du taux d'exploration (*exploration_rate*) sont facilement observables. Dans l'exemple donné, le taux d'exploration est gardé par défaut et n'apparait donc pas dans la définition du modèle.

```
model = DQN(
    "MlpPolicy",
    env,
    learning_rate=0.01,
    verbose=1,
    batch_size=64,
    train_freq=16,
    target_update_interval=2000,
    learning_starts=100,
    buffer_size=10000,
    tensorboard_log="./tb_logs/",
)
```

Figure 4 : Définition du modèle et de l'algorithme d'apprentissage. L'environnement Gym défini est associé à ce modèle.

IV.3/ *Démarrer l'apprentissage*

Nous pouvons alors démarrer la phase d'apprentissage. Pour cela, il est nécessaire de démarrer la simulation, comme présenté lors de l'installation d'Unreal Engine et d'AirSim.

Le script *dqn_car.py* est ensuite exécuté en s'assurant que la variable *load* est initialisée à *False*. Un nouveau modèle est alors créé et la phase d'apprentissage est lancée pour un nombre d'étape définie, comme présenté figure 5. Les meilleurs modèles sont régulièrement enregistrés et pourront ensuite être utilisés lors de la phase d'inférence.

```
model.learn(
    total_timesteps=100000,
    tb_log_name="dqn_airsim_car_run_" + str(time.time()), **kwargs
)
```

Figure 5 : Définition de la phase d'apprentissage sur 100 000 étapes. Les performances sont enregistrées à l'aide de tensorboard

IV.4/ *Optimiser l'apprentissage*

Afin d'accélérer l'apprentissage, il est possible d'accélérer le temps en simulation. Cela permet de réduire fortement le temps nécessaire pour entrainer un modèle en le réduisant à moins de deux heures (en vitesse x5). Pour cela, dans le fichier de configuration AirSim, placé dans les documents utilisateurs, permet de régler le paramètre « *ClockSpeed* » qui est pris en compte lors du lancement de la simulation. La même valeur doit être précisée au début du fichier Python *envs/car_env.py* avant le début de l'entrainement du modèle. Il est important de noter que le modèle n'est pas impacté directement par l'accélération du temps. Il faudra cependant régler en conséquence le nombre d'actions par secondes.

Le réglage de la variable *load* a True permet de charger un modèle déjà partiellement entrainé et de continuer l'apprentissage. Cette option est particulièrement utile lors de l'utilisation de ce système en séance d'activité pratique, permettant le pré-entraînement d'un modèle avant la séance puis la finalisation de l'entrainement en 15 minutes durant la séance. Si le modèle initial n'est pas trop entrainé, les performances seront significativement améliorées lors de cette deuxième phase, permettant alors aux élèves d'observer simplement l'évolution du comportement de l'agent. Entre les deux phases d'apprentissage, il est également possible de modifier la définition des récompenses et d'observer alors rapidement leur effet. En revanche, un changement des hyperparamètres, de l'algorithme d'apprentissage ou de la politique (tous définis lors de l'initialisation du modèle), demande de créer un nouveau modèle qui doit reprendre l'entraînement depuis le début.

V/ *Observer les résultats*

V.1/ *Utilisation d'un circuit de test*

L'évolution de la voiture sur le circuit peut être visualisée au cours de l'entrainement. Cela permet d'avoir un premier aperçu des performances de la voiture mais cela n'est pas suffisant.

Question 4 :

L'observation des performances de la voiture sur le circuit d'entrainement est-elle suffisante ? Quel est l'intérêt d'utiliser un autre circuit pour tester le modèle ?

Réponse 4 :

La visualisation des performances sur le circuit d'entrainement n'est pas représentative des performances réelles de l'agent. Nous pouvons être dans le cas du problème de surapprentissage, ayant alors un véhicule très performant sur des données connues (le circuit d'entrainement) mais échouant sur des données nouvelles (un nouveau circuit). Il est alors important d'utiliser un **circuit de test,** sur lequel le véhicule ne s'est pas entrainé, pour évaluer les performances de l'agent.

Les circuits utilisés pour notre exemple, et fournis comme exemples, sont présentés figure 6.

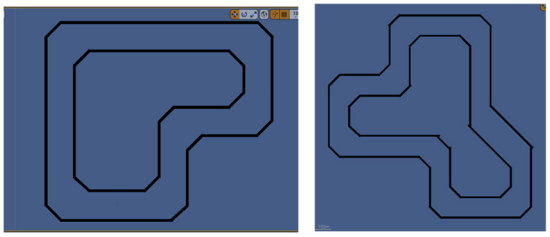


Figure 6 : Circuits utilisés : circuit d'entraînement à gauche et de test à droite. Les deux circuits doivent être différents pour évaluer la capacité de généralisation de l'agent mais doivent tout de même garder des caractéristiques similaires.

Afin d'observer l'évolution de l'agent sur l'un des circuits, il faut lancer la simulation du circuit correspondant puis lancer le script *dqn_car_test.py*. Dans ce dernier, il est nécessaire de préciser quel modèle utiliser et donc mettre le chemin d'accès au modèle enregistré lors de l'entrainement.

Ce script commence par réaliser plusieurs passages sur le circuit afin d'obtenir la récompense moyenne associée au modèle. Le véhicule se déplace en simulation ce qui donne également un aperçu des performances de l'agent.

Une vidéo présentant la conduite sur le circuit de test par un modèle pré-entraîné fourni est présentée sur le site Culture Science de l'Ingénieur[3] (accessible par le lien en pied de page ou en scannant le QR code présent sur cette page).

[3]-https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/apprentissage-par-renforcement-dela-conduite-dun-vehicule-sur-airsim

V.2/ *Visualiser une trajectoire pour un modèle*

Le même script affiche ensuite la trajectoire de la voiture afin de donner un aspect graphique permettant d'évaluer les performances. La figure 7 présente un exemple de ces trajectoires pour un modèle entrainé en un peu moins de deux heures.

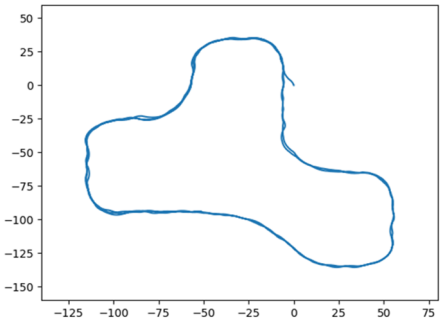


Figure 7 : Exemple d'une trajectoire obtenue sur le circuit de test pour un agent entrainé pendant environ 2h

La comparaison des trajectoires à différentes phases de l'entrainement permet d'observer simplement l'apport de l'apprentissage par renforcement dans le cadre de la conduite autonome.

VI/ Conclusion

Dans cet article, nous avons présenté un cas d'application concret d'un problème d'apprentissage par renforcement. Cette activité illustre l'apprentissage de la conduite autonome en comparant des agents avant et après la phase d'entrainement. L'effet des hyperparamètres peut également être étudié en changeant leur valeur dans la définition du modèle. Des modèles pré-entrainés permettent d'accélérer la phase d'apprentissage et de permettre ainsi de réaliser cette étude lors d'une séance de travaux pratiques.

Les méthodes présentées dans cet article sont adaptables simplement à d'autres problèmes d'apprentissage par renforcement, si l'on dispose d'un environnement simulé. Il est encouragé d'appliquer ces méthodes à de nouveaux problèmes afin d'observer le vaste champ d'application de l'apprentissage par renforcement. :

VII/ Références bibliographiques

[1] Documentation Stable Baselines 3 et Gym, https://stable-baselines3.readthedocs.io, https://www.gymlibrary.ml/

[2] Documentation AirSim, https://microsoft.github.io/AirSim/

[3] V. Mnih et al., Playing Atari with Deep Reinforcement Learning, arXiv, https://arxiv.org/abs/1312.5602

[4] V. Mnih et al., Human-level control through deep reinforcement learning, *Nature* **518,** 529–533 (2015), https://doi.org/10.1038/nature14236

Site Culture Science de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/apprentissage-par-renforcement-dela-conduite-dun-vehicule-sur-airsim

QR Code : Flashez-moi pour aller directement sur le site

INTEGRER LES CONNAISSANCES PHYSIQUES DANS LES RESEAUX DE NEURONES : APPLICATION A L'APPRENTISSAGE DE LOIS DE COMPORTEMENT DE MATERIAUX A PARTIR DE MESURES DE DEFORMATION PAR FIBRES OPTIQUES

ANTOINE BENADY[1], LUDOVIC CHAMOIN[1], EMMANUEL BARANGER[1]
[1] : *Université Paris-Saclay, CentraleSupélec, ENS Paris-Saclay, CNRS, LMPS - Laboratoire de Mécanique Paris-Saclay, 91190, Gif-sur-Yvette, France.*
{antoine.benady, ludovic.chamoin, emmanuel.baranger}@ens-paris-saclay.fr

Résumé *: Le travail présenté vise à utiliser des réseaux de neurones pour l'apprentissage de lois de comportement en mécanique des structures. Les réseaux de neurones classiques (au sens où il ne sont pas informés par la physique) présentent les inconvénients de nécessiter des volumes de données importants pour être entrainés, d'avoir des problèmes de généralisation à de nouvelles données et de manquer de garanties en termes d'interprétabilité. Cet article présente différentes façons de tenir compte de la physique dans l'apprentissage afin de pallier ces problèmes, ainsi qu'une stratégie suivie dans le cadre d'un projet de recherche en cours.*

Mots clés *: réseaux de neurones, jumeaux numériques, comportement mécanique des matériaux.*

I/ Introduction

L'endommagement des structures mécaniques est une préoccupation permanente en ingénierie, liée à des problématiques de durabilité et de sécurité. La thématique est actuellement l'objet de diverses activités de recherche ; un exemple caractéristique est le projet ERC DREAM-ON (site web : https://erc-dreamon.ens-paris-saclay.fr) dans lequel se place ce travail, qui s'intéresse aux structures mécaniques complexes et vise à aborder les challenges numériques liés au contrôle santé intégré de structures à grande échelle, afin d'aller des matériaux intelligents vers les structures intelligentes, capables de surveiller leur état de façon autonome, de prédire leur évolution à l'aide d'un jumeau numérique et de fonctionner de façon sûre même en mode dégradé (figure 1).

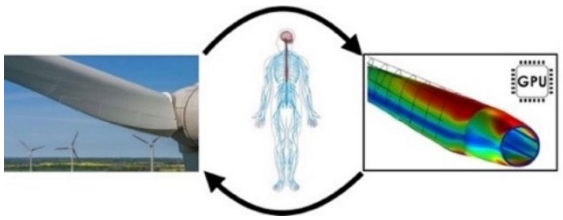


Figure 1 : contrôle santé d'une structure complexe avec un jumeau numérique.

Des fibres optiques immergées dans la structure à surveiller sont utilisées pour mesurer les déformations dans le matériau (figure 2). L'idée est de créer un jumeau numérique hybride, combinant les modèles physiques (qui représentent une riche histoire des sciences de l'ingénieur, et qui fournissent une forte connaissance a priori) et les techniques d'apprentissage issues de l'IA (ici, les réseaux de neurones), pour le diagnostic, la prédiction et la prise de décision.

Dans le cas d'application visé, on souhaite apprendre des lois de comportement de matériaux grâce aux réseaux de neurones entrainés avec les mesures issues de fibres optiques, sans oublier des siècles de connaissances physiques : cet article présente les différentes façons d'intégrer l'a priori physique dans les réseaux de neurones. Dans une première partie, il est question des architectures de réseaux guidées par la physique. Une deuxième partie présente l'entraînement de réseaux avec une fonction coût physique. Dans un troisième temps, l'initialisation du réseau avec un a priori physique est traitée. Enfin on montre qu'il est possible d'utiliser les réseaux de neurones pour apprendre l'erreur d'un modèle physique. Certaines parties seront illustrées par le cas d'application visé. Bien entendu, il est possible de coupler ces techniques entre elles.

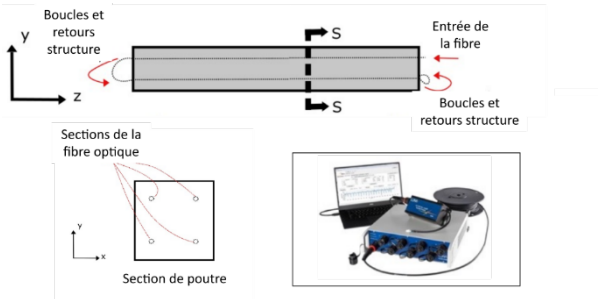


Figure 2 : Cas d'étude sur une poutre en béton et dispositif de mesure par fibre optique (système ODiSI 6108 de LUNA en bas à droite).

II/ Architecture du réseau guidée par la physique

Les réseaux de neurones classiques sont réputés pour ne pas offrir de garanties lors de la phase de prédiction, cela étant lié à la faible interprétabilité des résultats. Une façon d'imposer des contraintes physiques dans le réseau est de choisir une architecture qui garantit certaines propriétés. Par exemple, les couches de convolution des CNN (réseaux de neurones convolutifs, souvent utilisés en traitement d'images) présentent l'intérêt de garantir une invariance par translation des résultats.

Une autre possibilité, plus orientée vers l'interprétabilité que vers les garanties de prédiction, consiste à interpréter des grandeurs intermédiaires du réseau comme des quantités physiques [5], les grandeurs intermédiaires pouvant intervenir dans la fonction coût.

Dans le cadre du projet ERC, une architecture garantissant le respect de la thermodynamique des milieux continus est proposée. Le réseau prend en entrée une déformation (grandeur cinématique) et donne en sortie une contrainte (grandeur d'effort), traduisant le comportement du matériau. Un potentiel thermodynamique, permettant de définir la relation de comportement, est utilisé comme grandeur intermédiaire. L'architecture utilisée est inspirée de [4] mais présente des différences, notamment au niveau de la convexité du potentiel par rapport à la déformation [1] (propriété traduisant le respect du second principe de la thermodynamique). La figure 3 présente un schéma de l'architecture utilisée. L'opération de passage du potentiel thermodynamique à la contrainte est effectuée par différentiation automatique, une technique permettant le calcul de dérivées partielles (utile notamment pour la rétropropagation), et qui est déjà implémentée dans les bibliothèques classiques de Deep Learning (PyTorch, Tensorflow).

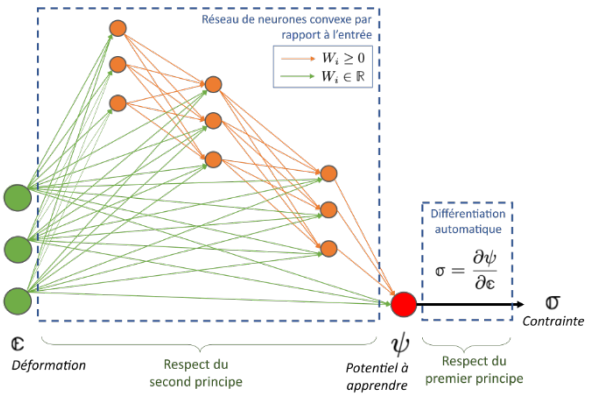


Figure 3 : Architecture utilisée pour apprendre des lois de comportement vérifiant les principes thermodynamiques, avec W_i les poids du réseau.

III/ Utilisation d'une fonction coût physique

Il est également possible de pénaliser le non-respect d'une équation physique en intégrant son résidu dans la fonction coût utilisée pour l'entrainement du réseau. Cette idée a initialement été proposée dans [6] dans le cas particulier de l'apprentissage de l'équation de Navier-Stokes par un réseau de neurones. Pour cela, une fonction coût avec un terme d'écart aux données et un terme de non-respect de l'équation physique est introduite :

$$\mathcal{L} = \mathcal{L}_{\text{données}} + \mathcal{L}_{\text{physique}}$$

$$\mathcal{L}_{\text{données}} = \frac{1}{N_{obs}} \sum ||\mathbf{Y}_{\text{reseau de neurones}} - \mathbf{Y}_{\text{observations}}||^2$$

$$\mathcal{L}_{\text{physique}} = \frac{1}{N_{pts}} \sum |\ \text{f(u(x,t))}|^2$$

avec f(u(x, t)) le résidu de l'équation physique portant sur la grandeur d'intérêt u aux N_{pts} points d'évaluation (x, t). On note que les N_{obs} points d'évaluation du terme d'écart aux données ne sont pas nécessairement les mêmes que ceux du terme de non-respect de l'équation physique.

La méthode mise en place dans le cadre du projet ERC pour apprendre des lois de comportement s'inspire de cette structure de fonction coût, avec l'utilisation de la fonctionnelle « erreur en relation de comportement modifiée » (mCRE) pour entrainer le réseau de la figure 3. Cette fonctionnelle, utilisée depuis les années 90 en mécanique dans le contexte de recalage de paramètres [2, 3], se décompose en 2 termes :

- un terme d'écart aux observations ;
- un terme d'erreur de modèle fondé sur l'erreur en relation de comportement. Ce terme quantifie le niveau de non-vérification des relations de comportement pour un couple déplacement-contrainte admissible (vérifiant l'équilibre et les conditions aux limites) et fait intervenir le potentiel ψ recherché.

IV/ Initialisation du réseau par la physique

Un problème couramment rencontré lors de l'entraînement d'un réseau de neurones concerne la quantité de données nécessaire d'autant plus lorsqu'on souhaite apprendre à partir de mesures expérimentales. Une façon d'augmenter la quantité de données à disposition consiste à procéder à un entraînement du réseau en deux temps. Dans un premier temps, un entraînement est réalisé sur des données synthétiques qui sont générées à partir d'un modèle physique a priori, supposé proche de la réalité. A l'issue de ce premier entraînement, le réseau est initialisé proche de la réponse observée dans les données expérimentales et un deuxième entraînement, cette fois-ci avec les données mesurées (en quantité moindre) permet de converger

vers un modèle recalé. La démarche est détaillée dans la figure 4.

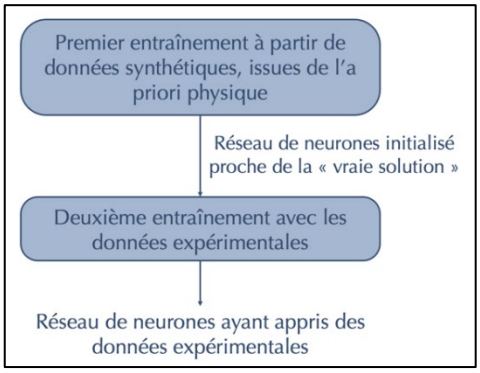


Figure 4 : prise en compte de l'a priori physique dans l'initialisation du réseau.

V/ Réseau de neurones utilisé pour apprendre l'erreur du modèle physique

Enfin, on trouve aussi dans la littérature l'utilisation de réseaux de neurones pour apprendre l'erreur d'un modèle physique. Si on s'intéresse à une grandeur d'intérêt $\boldsymbol{u}$, on peut la décomposer comme suit :

$$\boldsymbol{u} = \boldsymbol{u_{phy}} + \boldsymbol{u_{NN}}$$

avec $\boldsymbol{u_{phy}}$ la grandeur prédite par un modèle physique (qui par définition ne fait que modéliser une partie de la réalité) et $\boldsymbol{u_{NN}}$ le complément prédit par un réseau entrainé de manière à minimiser l'écart à des données mesurées.

Par exemple, si on souhaite apprendre une loi de comportement avec un endommagement générant une rigidité du matériau différente en traction et en compression (ce qui se traduit par le potentiel en rouge sur la figure 5, correspondant à une parabole définie par morceaux), une possibilité est de supposer la loi de comportement élastique linéaire (parabole noire sur la figure 5) et de faire apprendre le complément c'est-à-dire la non-linéarité au réseau de neurones.

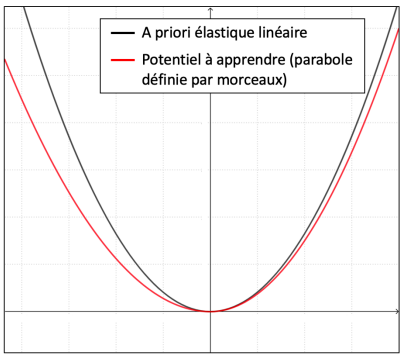


Figure 5 : Potentiel en fonction de la déformation pour un modèle supposé élastique linéaire, et pour un modèle avec endommagement.

VI/ Références bibliographiques

[1] Amos, Brandon and Xu, Lei and Kolter, J. Zico, Input Convex Neural Networks, https://arxiv.org/abs/1609.07152 ; (2016)

[2] Ladevèze, P., Nedjar, D., and Reynier, M. Updating of finite element models using vibration tests. AIAA Journal, 32(7) :1485–1491. (1994)

[3] Ladevèze, P. A modelling error estimator for dynamical structural model updating. In Elsevier, editor, Advances in Adaptive Computational Methods in Mechanics. P. Ladevèze, J.T. Oden eds. (1998)

[4] Masi F, Stefanou I, Vannucci P, Maffi-Berthier V, Thermodynamics-Based Artificial Neural Networks for Constitutive Modeling, Journal of the Mechanics and Physics of Solids 147 :104277 (2021)

[5] Nikhil Muralidhar, Jie Bu, Ze Cao, Long He, Naren Ramakrishnan, Danesh Tafti, and Anuj Karpatne. Physics-Guided Deep Learning for Drag Force Prediction in Dense Fluid-ParticulateSystems. Big Data, 8(5) :431–449, October 2020.

[6] Raissi M, Perdikaris P, Karniadakis G.E., Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations, Journal of Computational Physics 378 : 686 707 (2019)

DETECTION ET CLASSIFICATION AUTOMATIQUE DES DOCUMENTS POUR L'APPLICATION KYC

WILLIAM KETCHANTANG [1], DIMITRIOS TSOLAKIDIS [2], KEVIN MEETOOA [2]
[1] : PhD, TEAM LEADER MACHINE LEARNING AND COMPUTER VISION CHEZ VIALINK
[2] : MACHINE LEARNING AND COMPUTER VISION ENGINEER CHEZ VIALINK
william.ketchantang@vialink.fr, dimitrios.tsolakidis@vialink.fr, kevin.meetooa@vialink.fr

Résumé : *Cet article présente la détection et la classification automatique de documents scannés ou pris en photos pour des fins de digitalisation de l'entrée en relation clients (KYC : Know Your Customer). Nous présentons un modèle de Deep Learning de détection et de classification automatique des documents capable de détecter des documents sur une image avec une efficacité de 90 % et un temps moyen de l'ordre de la seconde sur des machines équipées de cpu i7 standard.*

Mots clés *: Détection, Classification, Deep Learning, Documents, KYC*

I Introduction

I.1 *Application KYC*

Notre société se digitalise de plus en plus, et à ce titre, nous souhaitons aller plus loin en permettant l'ouverture des comptes bancaires, la souscription de prêts de consommation, immobiliers 100 % en ligne par exemple. Pour cela, nous avons besoin d'identifier le souscripteur et de contrôler automatiquement ses documents, afin de s'assurer que tout le processus ne souffre d'aucune irrégularité. Ce processus s'appelle couramment KYC (« *Know Your Customer* »), consistant à digitaliser l'entrée en relation. Il peut être schématisé comme l'illustre la Figure 1. Ce processus 100 % numérique permet un gain d'efficacité chez les opérationnels, améliore la qualité de vie des particuliers, et permet la conformité de la réglementation bancaire sur la lutte contre les fraudes et le blanchiment d'argent.

Le processus KYC est un processus complexe. En début de processus, il est important de détecter et classifier automatiquement les documents transmis par l'utilisateur. Ces documents sont variables selon les contrôles souhaités par les clients, à l'instar des pièces d'identité, justificatifs de domicile, avis d'imposition, bulletins de salaire, Kbis, etc... Dans cet article, nous décrirons le module de détection/classification automatique des documents, en charge de localiser précisément tout type de documents supportés, y compris leur faces avant et arrière, indépendamment des conditions d'acquisition (camera, scanner, luminosité ambiante non contrôlée, déformations géométriques des objets dans l'image, .etc). Comme dans l'état de l'art, nous allons utiliser le mot Détection au lieu de Détection et Classification.

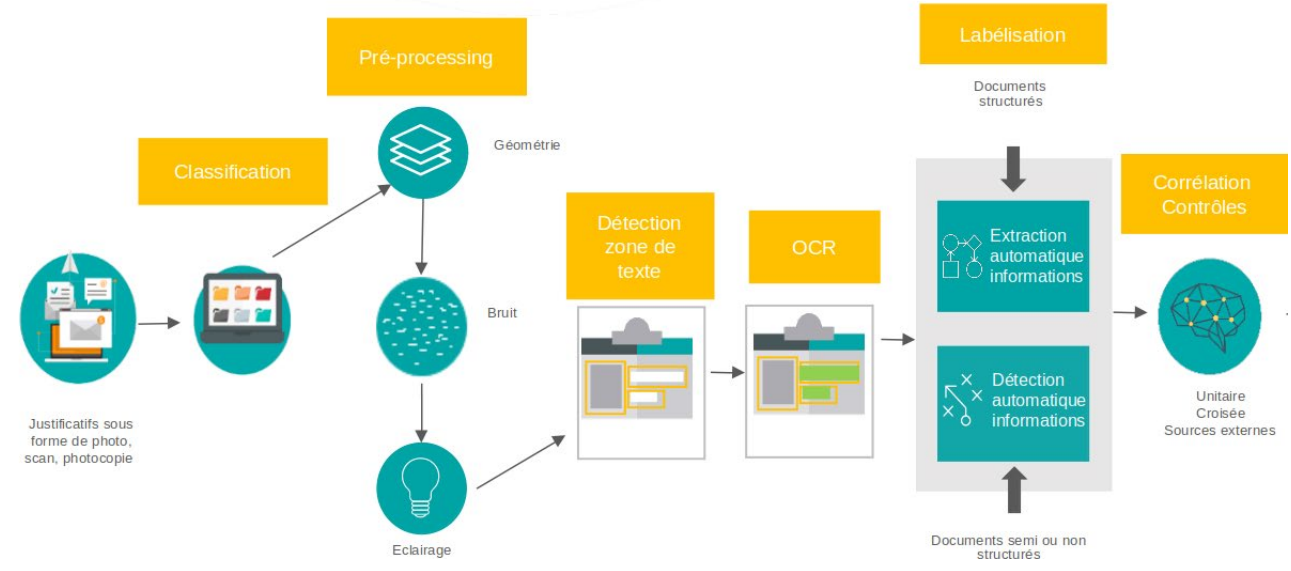


Figure 1 : Aperçu d'un processus KYC

I.2 *Détection des documents*

La détection d'objets est l'une des tâches les plus importantes et les plus difficiles de la vision par ordinateur, utile pour la vidéosurveillance, la conduite autonome, l'identité numérique, etc. L'objectif est de localiser tous les objets et leurs catégories respectivement avec des scores de confiance. Cette tâche est bien connue et bien traitée dans l'état de l'art. Nous identifions deux groupes d'approches : D'une part, les premières approches sont basées sur des **méthodes traditionnelles** qui nécessitent beaucoup d'efforts de « *Feature Engineering* ». D'autre part, les nouvelles approches dites modernes se concentrent sur les **méthodes d'apprentissage profond**.

Les méthodes traditionnelles peuvent être découpées en deux groupes. Tout d'abord, certaines approches sont basées sur l'extraction de points clés des objets que nous voulons détecter. L'auteur [4] utilise SURF (« *Speed Up Robust Features* », [2]) et BoW (« *Bag of Words* ») pour extraire les caractéristiques des points clés des objets. Ces caractéristiques sont ensuite utilisées pour entraîner un classifieur SVM (Machine à vecteurs de support) afin de détecter les catégories d'objets. Les auteurs obtiennent de bons résultats (environ 70 % d'efficacité sur des bases d'images publiques Caltech [5]). Cependant, extraire les caractéristiques des nouvelles catégories d'objets nécessite de revisiter la méthode d'extraction, d'où un temps de développement plus important.

De plus, certains auteurs utilisent d'autres approches d'extraction de caractéristiques n'utilisant pas les points clés, comme le LBP (« *Local Binary Pattern* », [20]) utilisé par [21] pour extraire les caractéristiques des visages, pour des fins de détection. L'histogramme des gradients est également utilisé pour les caractéristiques des objets, comme décrit par [17], pour des fins de détection.

Toutes les approches basées sur l'extraction de caractéristiques de type « *Feature Engineering* » nécessitent un effort important de la part des scientifiques pour construire les algorithmes ad-hoc permettant d'extraire les caractéristiques pertinentes des objets à détecter. Si nous voulons détecter une nouvelle catégorie d'objets, nous devons revisiter l'algorithme d'extraction de caractéristiques pertinentes, et y passer beaucoup de temps, sans garantie de résultats à la hauteur des attentes. Il s'agit d'une limite importante des approches dites de « *Feature Engineering* ». C'est pourquoi, de plus en plus de scientifiques utilisent maintenant des approches d'apprentissage profond pour espérer obtenir des caractéristiques pertinentes en peu de temps et avoir une très bonne généralisation sur les nouveaux types d'objets.

Plusieurs auteurs [8] utilisent des approches d'apprentissage profond pour détecter des objets. Deux types de détecteurs sont détaillés : Le premier détecteur est basé sur les réseaux de proposition de régions sous forme de boîte rectangulaire, avec la catégorie associée (R-CNN « *Region Convolutional Neural Network* »). La précision de tels réseaux est de l'ordre de 75 % sur la base d'images publique Pascal VOC 2007 [22].

Le second détecteur proche du précédent mais plus léger (YOLO « *You Only Look Once* ») est également basé sur une proposition de régions sous forme de boîte englobante et la catégorie associée de l'objet [12]. Le temps de traitement est plus faible que celui du premier détecteur, de plus ce modèle tourne facilement sur les plateformes embarquées de type téléphone mobile. Cependant, la précision de la localisation est inférieure (63%) à celle de la première approche sur la base d'images publiques Pascal VOC 2007.

Avec les approches d'apprentissage profond, il est très important d'avoir une base d'images représentative des objets et de laisser le modèle apprendre les caractéristiques pertinentes des objets, tout en étant capable de généraliser la caractérisation des objets qui ne sont pas présents dans notre jeu de données d'apprentissage. Et cela, sans aucun effort supplémentaire de la part des scientifiques. C'est l'avantage d'utiliser des approches d'apprentissage profond pour la détection d'objets. Les auteurs [11] résument également les avantages des approches d'apprentissage profond par rapport aux approches traditionnelles. Dans notre cas, où nous avons plusieurs documents de différents types, structurés et non structurés, les approches d'apprentissage profond sont préférées.

Cet article est structuré comme suit : Dans la section suivante, nous décrivons dans un premier temps l'architecture du modèle que nous utilisons. Nous présentons dans un second temps les expériences et les résultats. Enfin, nous donnons la conclusion et les perspectives.

II *Modèle de Détection des documents*

II.1 *Architecture*

Nous utilisons une architecture U-Net, introduite pour la première fois par [13]. Le modèle de segmentation U-Net a montré de meilleures performances que les réseaux purement convolutifs. Il s'agit d'un réseau en forme de U composé d'un encodeur et d'un décodeur, comme l'illustre la Figure 2.

Intuitivement, l'encodeur apprend à extraire les éléments caractéristiques discriminants des objets. L'encodeur, quant à lui, réduit les dimensions spatiales en sortie de chaque couche tout en augmentant le nombre de filtres. Le tenseur obtenu en sortie de l'encodeur est généralement appelé goulot d'étranglement ou espace latent. On parle également de vecteur de caractéristiques.

Le décodeur apprend à localiser le document. Le point d'entrée du décodeur est le vecteur de caractéristiques de l'encodeur. Le décodeur augmente les dimensions spatiales en sortie de chaque couche tout

en réduisant le nombre de filtres, jusqu'à obtenir la résolution spatiale de l'image d'entrée.

La « *skip-connection* » permet d'ajouter la sortie de l'encodeur à une profondeur donnée du réseau à l'entrée du décodeur à la même profondeur. Ainsi, le sur-échantillonnage dans la branche du décodeur garde les détails de l'encodeur. Par conséquent, la sortie du décodeur à chaque profondeur du réseau est de meilleure qualité.

En sortie du modèle, les dimensions spatiales du masque prédit sont les mêmes que les dimensions de l'image d'entrée, ce qui permet de faire une prédiction de la catégorie de chaque pixel de l'image d'entrée.

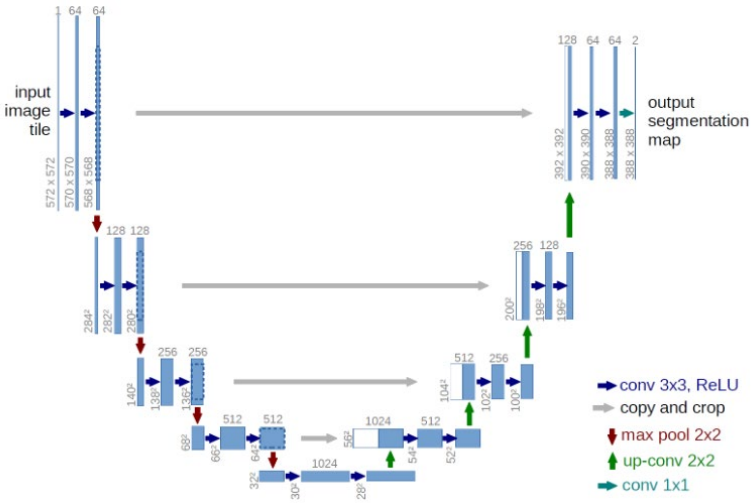


Figure 2 : Architecture U-Net.

Chaque boîte bleue correspond à une carte de caractéristiques multicanaux. Le nombre de canaux est indiqué en haut de la boîte et la taille x-y de la boîte est indiquée sur le bord inférieur gauche de celle-ci. Les boîtes blanches représentent des cartes de caractéristiques copiées issues de la partie encodeur. Les flèches indiquent les différentes opérations [13].

Nous utilisons un modèle de type EfficientNet-B3 dérivé de la famille EfficientNet [18] comme encodeur de la branche U-Net.

II.2 *Fonction de coût*

Nous utilisons la fonction de coût « *Dice* » [16] et la fonction de coût dite « *focale* » [10] pour l'entraînement du modèle. L'idée principale est d'assigner à chaque pixel une classe d'appartenance. Ainsi, nous avons un masque prédit pour chaque classe de documents. La fonction de coût *Dice* mesure donc le chevauchement relatif entre un masque de classe de document prédit et sa vérité terrain. La formule de la fonction de coût de *Dice* est présentée ci-dessous :

$$D_{loss} = 2\frac{\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

où p_i est la valeur d'un pixel prédit pour une classe de document donnée (0 ou 1) et g_i est la valeur du même pixel issu de la vérité de terrain.

La fonction de coût *Dice* n'est pas adaptée au déséquilibre de classes. Pour cette raison, nous utilisons une fonction de coût dite « focale » en plus de la fonction de coût *Dice*. La fonction de coût focale est exprimée par la formule suivante :

$$F_{loss}(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t)$$

où p_t correspond à la probabilité d'appartenance d'un pixel à la bonne classe. γ contrôle la forme de la courbe. Plus la valeur de γ est élevée, plus les pixels mal classés sont mis en évidence dans le calcul de la fonction de coût. Ainsi, cette fonction de coût est plus adaptée lorsque les classes sont déséquilibrées.

Nous obtenons la fonction de coût finale en additionnant les deux fonctions de coût :

$$Total_{loss} = D_{loss} + F_{loss}$$

II.3 *Entraînement*

Les images d'entrée et leurs masques de vérité de terrain correspondantes sont utilisées pour entraîner le réseau. Les images d'entrée et les masques sont redimensionnés en taille 256 x 256 pixels. La base de documents se compose de plusieurs classes, avec 12370 images de documents au total. Dans cette base de documents, nous avons également des documents augmentés artificiellement (rotation, changement de luminosité, distorsion, etc...).

Le réseau a été entraîné :

- Pendant 40 époques
- Avec une taille de batch de 4
- En utilisant l'optimiseur Adam [9]
- Avec un taux d'apprentissage de 0,001.
- Avec des poids initiaux issus du pré-entraînement du modèle sur la base d'images publiques ImageNet [15].

Pour la fonction de coût focale, nous utilisons les meilleures valeurs mentionnées par les auteurs [10] avec $\alpha = 0.25$et $\gamma = 2$.

Les résultats de l'entraînement sont illustrés par les fonctions de coût de la figure 5.

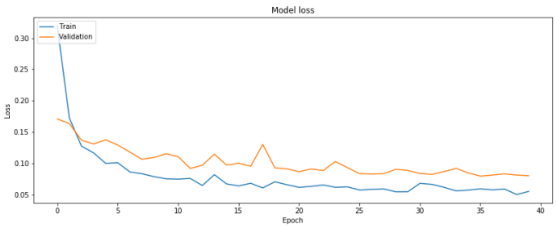


Figure 5 : Évolution de la fonction de coût D_loss durant l'entraînement

Nous observons que la fonction de coût converge aussi bien sur la base de documents d'entraînement que sur la base de documents de validation. Ainsi, notre modèle est bien entraîné.

III Résultats expérimentaux

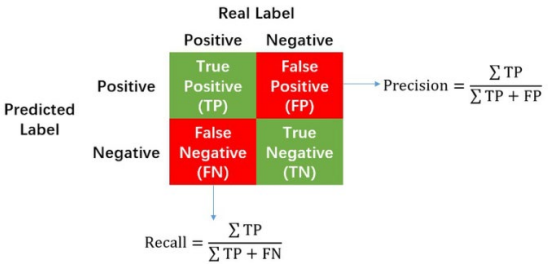
Nous utilisons trois métriques pour évaluer l'efficacité de notre modèle : Précision, Rappel et IoU.

Pour un document donné et deux classes binaires, nous définissons :

- **TP** comme le nombre de vrais positifs : C'est le nombre de pixels de la classe positive qui ont été correctement prédits.
- **FP** comme le nombre de faux positifs : C'est le nombre de pixels de la classe négative qui ont été prédits à tort comme positifs.
- **FN** est le nombre de faux négatifs : Il s'agit du nombre de pixels de la classe positive qui ont été prédits à tort comme négatifs.

Par conséquent,

- La **précision** quantifie le nombre de prédictions correctes de pixels pour la classe positive.
- Le **rappel** quantifie la proportion de prédictions de pixels positifs par rapport au nombre total de pixels positifs dans le document.



Cette définition peut ensuite être étendue à plus de deux classes. Pour un document donné et deux classes binaires, nous définissons IoU comme suit :

$$IoU = \frac{P \cap GT}{P \cup GT}$$

Où :

- P correspond au masque prédit pour la classe positive
- GT correspond au masque de vérité pour la classe positive.

Au numérateur, nous calculons la zone de chevauchement entre le masque prédit et le masque de vérité.

Le dénominateur est l'union entre le masque prédit et le masque de vérité.

En divisant la zone de chevauchement par la zone d'union, nous obtenons le score final d'Intersection sur Union (IoU).

Figure 6 : Vert: vérité de terrain du masque. Rouge : masque prédit.

Cette définition peut également être étendue à plus de deux classes.

Les mesures de précision, de rappel, et de IoU score obtenues sur notre base de tests sont présentées dans le tableau 1.

Précision	Rappel	IoU
92.71%	94.14%	**88.23%**

Tableau 1 : Efficacité du modèle sur notre base de test de documents privés.

Avec 92 % de précision, 94 % de rappel, et 88 % de IoU score, nous considérons que le modèle utilisé donne des résultats satisfaisants.

De plus, notre modèle est capable d'atteindre un temps d'inférence de 1,25 secondes en moyenne sur une machine équipée d'un CPU i7, ce qui est satisfaisant pour nos besoins.

Nous présentons quelques résultats qualitatifs obtenus sur quelques documents de notre jeu de données de test, à la figure 7.

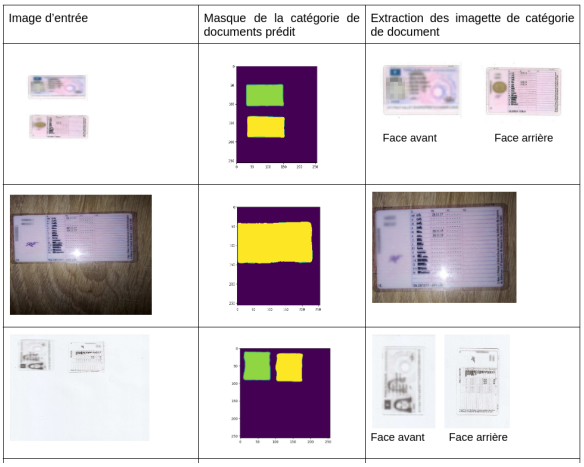


Figure 7 : Illustration de la détection et classification des documents. Vert : Face recto du permis de conduire français. Jaune : Face verso du permis de conduire français.

Comme nous pouvons le voir ci-dessus, le modèle est robuste et permet de localiser avec précision les documents, quelle que soit leur qualité. Cela s'explique par le fait que nous avons entraîné le modèle sur un grand ensemble de documents comprenant des documents de bonne et de mauvaise qualité.

IV Conclusion

Nous avons utilisé un modèle de détection de documents dans une image basée sur l'architecture U-Net. L'efficacité (IoU score) du modèle sur notre base de 12370 documents internes de diverses catégories (Cartes d'identités, passeports de différents pays, .etc) est de 90 % environ. Le temps d'inférence sur une infrastructure classique équipée de CPUs i7 standards, est de l'ordre de la seconde pour 1 image.

Pour la suite, nous augmenterons la taille de notre base avec plus de catégories de documents. Ainsi, notre modèle supportera encore plus de catégories en production, et généralisera encore mieux. De plus, nous envisageons changer le l'encodeur « *backbone* » par un encodeur de type ResNet par exemple, plus efficace mais plus lourd.

V Bibliographie

[1] Agarap, Abien Fred. Deep Learning using Rectified Linear Units (ReLU). 2018, https://arxiv.org/pdf/1803.08375.pdf.

[2] Bay, Herbert, et al. "SURF: Speed Up Robust Features." ECCV, 2006.

[3] Chollet, Francois. "Xception: Deep Learning with Depthwise Separable Convolutions}." 2017, https://arxiv.org/pdf/1610.02357.pdf

[4] Farooq, Javeria. "Object Detection and Identification using SURF and BoW Model." ResearchGate, April 2016.

[5] Fei-Fei, L., et al. "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories." IEEE. CVPR, 2004.

[6] G. Lowe, David. "Distinctive Image Features from Scale-Invariant Keypoints." IJCV, 2004.

[8] Jiao, Licheng, et al. "A Survey of Deep Learning-based Object Detection." IEEE, 11 July 2019.

[9] Kingma, Diederik, and Jimmy Ba. "Adam: A Method for Stochastic Optimization." 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[10] Lin, Tsung-Yi, et al. "Focal Loss for Dense Object Detection." IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2999-3007.

[11] Mahomy, Niall O. "Deep Learning vs. Traditional Computer Vision." Advances in Computer Vision Proceedings of the 2019 Computer Vision Conference (CVC), 30 october 2019.

[12] Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[13] Ronneberger, Olaf, et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, 2015, pp. 234–241.

[14] Rublee, Ethan, and Gary Bradski. "ORB: an efficient alternative to SIFT or SURF." IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision, 2011.

[15] Russakovsky, Olga, et al. "ImageNet Large Scale Visual Recognition Challenge." Int. J. Comput. Vis., vol. 115, 2015, pp. 211-252.

[16] Sudre, Carole, et al. "Generalized Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations." Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, 2017.

[17] Sultana, Marjia, et al. "Object Detection using Template and HOG Feature Matching." International Journal of Advanced Computer Science and Applications, July 2020

[18] Tan, Mingxing, and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019.

[19] Zhang, Zhilu, and Mert Sabuncu. "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels." NeurIPS (2018), 2018.

[20] Lizé, Jade, et al. "Local binary pattern and its variants: application to face analysis." HAL Id: hal-02924534, 28 August 2020.

[21] Ahonen, T., Hadid, A., Pietikäinen, M. (2004). Face Recognition with Local Binary Patterns. In: Pajdla, T., Matas, J. (eds) Computer Vision - ECCV 2004. ECCV 2004. Lecture Notes in Computer Science, vol 3021. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24670-1_36

[22] Mark Everingham and Luc Van Gool and C. K. I. Williams and J. Winn and Andrew Zisserman. "The PASCAL Visual Object Classes (VOC) challenge."

[23] https://paperswithcode.com/sota/object-detection-on-pascal-voc-2007

INTRODUCTION AUX METHODES D'ACCELERATION DE RESEAUX DE NEURONES

EDOUARD YVINEC [1], ARNAUD DAPOGNY [2], KÉVIN BAILLY [3]
[1] Doctorant à l'ISIR de Sorbonne Université et chercheur à Datakalab
[2] Chercheur à Datakalab
[3] Maitre de conférence à l'ISIR de Sorbonne Université et directeur de la recherche de Datakalab

Résumé *: Malgré des performances impressionnantes, le déploiement des réseaux de neurones profonds reste limité. Ceci s'explique en partie par le coût important (en terme de temps de calcul, d'espace mémoire et d'énergie) nécessaire à l'inférence. Cet article présente les méthodes de compression et d'accélération des réseaux de neurones au travers de trois grandes catégories : l'élagage, la décomposition et recomposition de couches et la quantification. Les performances de ces approches sont données pour des tâches de vision (classification, détection segmentation) avec des architectures état de l'art (ResNets, MobileNets et EfficientNets).*

I/ Introduction

L'objectif de l'apprentissage profond par réseau de neurones est de pouvoir effectuer des prédictions pertinentes dans le cadre d'une tâche donnée. On compte parmi ces tâches, entre autres, celles liées à la vision par ordinateur qui nous serviront d'exemples tout au long de cet article. En particulier, on trouve la classification [1], la détection d'objets dans une image [2] ou encore la segmentation sé- mantique [3]. Malgré les spécificités liées à chacune de ces tâches, les architectures des réseaux de neurones, utilisées pour obtenir les remarquables performances que l'on reconnaît aujourd'hui à l'apprentissage profond, présentent de nombreuses similarités dont, entre autres, la présence d'une base commune (souvent appelée backbone). Pour cette étude, nous allons restreindre le bestiaire des réseaux de neurones aux ResNets [4] (les plus utilisés), aux MobileNets [5] et aux EfficientNets [6] (ces deux derniers exemples étant connus pour leur caractère compact). Afin de réduire le coût du déploiement de tels modèles, en termes de consommation énergétique ainsi qu'en terme de coût des ressources de calculs, une large gamme de méthodes ont été développées. Dans cet article, nous nous intéresserons aux principales approches présentées dans la figure 1. Nous aborderons tout d'abord les approches par élagage (section 3) qui consistent à supprimer des neurones (élagage structuré) ou des opérations (élagage non structuré). Nous présenterons ensuite les méthodes par décomposition et recomposition des couches du réseau (section 4). Pour finir, nous présenterons l'approche par quantification qui vise à réaliser les opérations arithmétiques à virgule fixe sur un nombre restreint de bits (section 5).

II/ Notations et Formalisme

Soit F un réseau de neurones artificiels à L couches. L'essentiel des calculs effectués dans un réseau de neurones artificiels F se trouve dans les couches de convolutions et dans les couches densément connectées. Ces couches, ainsi que leurs variantes, (convolutions

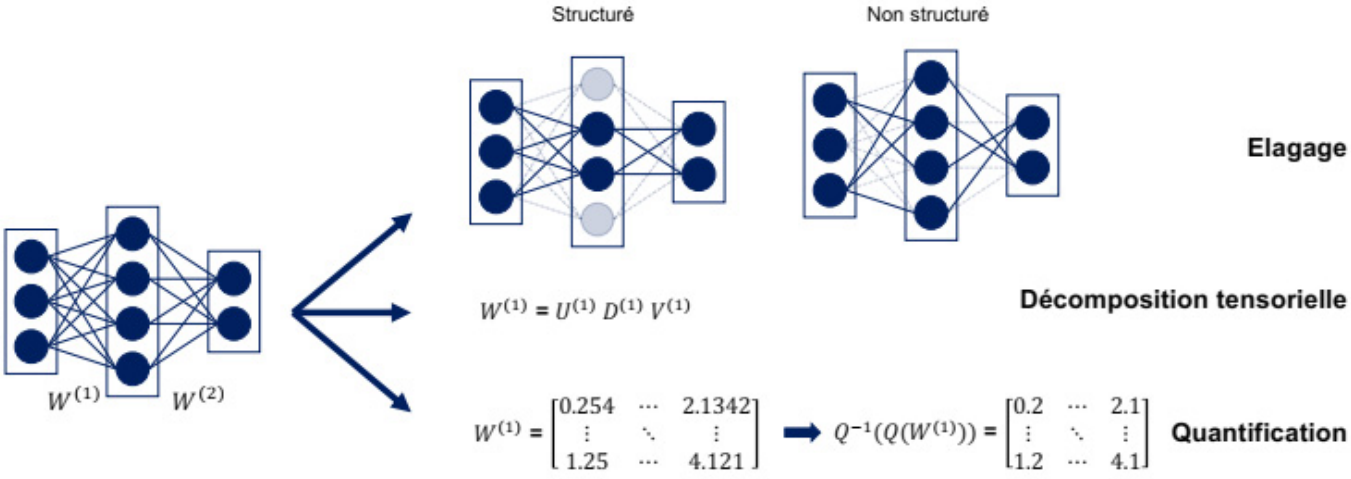


Figure 1 – Les 3 grandes catégories de méthodes de compression de réseaux de neurones présentées dans cet article

spatiales, convolutions à trous...) définissent une multiplication matricielle.

Soit f une telle couche, de fonction d'activation σ, on notera

$$f: \begin{matrix} \mathcal{A}^{d_e} \to & \mathcal{B}^{d_s} \\ X \mapsto & \sigma(WX+B) \end{matrix} \quad (1)$$

où d_e et d_s désignent respectivement les dimensions d'entrée et de sortie, $\mathcal{A}$ et $\mathcal{B}$ définissent chacun un $\mathbb{R}$ - espace vectoriel. Dans le cas d'une couche densément connectée on aura $\mathcal{A} = \mathbb{R}$ et dans le cas d'une couche convolutionelle sur une image 224 par 224 pixels on aura $\mathcal{A} = \mathbb{R}^{224\times224}$. On appellera W le tenseur de poids et B le vecteur de biais.

III/ Élagage (pruning)

La première catégorie de méthodes permettant d'accélérer les calculs effectués par un modèle F est l'élagage. Cela consiste à retirer des calculs afin de limiter le nombre d'opérations à réaliser. On oppose deux paradigmes et deux implémentations.

La Structure Les opérations supprimées du graphe peuvent présenter une cohérence structurelle ou non. En pruning structuré, les éléments supprimés sont de l'ordre du neurone (ou de la feature map dans le cas d'un réseau convolutif). En reprenant les notations précédentes, le pruning structuré revient à retirer des lignes ou des colonnes de W. Avant de décrire le pruning non-structuré, nous vous proposons de détailler un élément important du pruning structuré : *comment éliminer le biais ?*

Soit un réseau F à deux couches tel que

$$F: \quad X \mapsto W^{(2)}\sigma\left(W^{(1)}X + B^{(1)}\right) + B^{(2)} \quad (2)$$

Supposons que nous tenons pour certain que la meilleure option est de retirer la seconde ligne de la matrice W_1 et notons $\mathfrak{W}^{(1)}$, $\mathfrak{W}^{(2)}$, $\mathfrak{B}^{(1)}$, $\mathfrak{B}^{(2)}$ les paramètres du réseau élagué $\mathfrak{F}$. Pour le moment, on suppose que la seconde ligne de W_1 est simplement mise à 0. Dans ce cas, on ne peut pas simplement la retirer et retirer le biais correspondant sans changer la fonction de prédictions.

$$\mathfrak{F}(X) = W^{(2)}\sigma\left(\begin{pmatrix} W_{1,1}^{(1)} & \cdots & W_{1,N}^{(1)} \\ 0 & \cdots & 0 \\ W_{3,1}^{(1)} & \cdots & W_{3,N}^{(1)} \\ \vdots & \ddots & \vdots \\ W_{M,1}^{(1)} & \cdots & W_{M,N}^{(1)} \end{pmatrix}\begin{pmatrix} X_1 \\ \vdots \\ X_N \end{pmatrix} + \begin{pmatrix} B_1^{(1)} \\ \vdots \\ B_M^{(1)} \end{pmatrix}\right) + B^{(2)} \quad (3)$$

Pour déduire les poids élagués, nous allons propager le biais correspond au second neurone de la première couche. Ainsi, on obtient

$$\begin{cases} \mathfrak{F}(X) = \mathfrak{W}^{(2)}\sigma\left(\mathfrak{W}^{(1)}X + \mathfrak{B}^{(1)}\right) + \mathfrak{B}^{(2)} \\ \mathfrak{W}^{(1)} = \begin{pmatrix} W_{1,1}^{(1)} & \cdots & W_{1,N}^{(1)} \\ W_{3,1}^{(1)} & \cdots & W_{3,N}^{(1)} \\ \vdots & \ddots & \vdots \\ W_{M,1}^{(1)} & \cdots & W_{M,N}^{(1)} \end{pmatrix} \\ \mathfrak{W}^{(2)} = \begin{pmatrix} W_{1,1}^{(2)} & W_{1,3}^{(2)} & \cdots & W_{1,M}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{K,1}^{(2)} & W_{K,3}^{(2)} & \cdots & W_{K,M}^{(2)} \end{pmatrix} \\ \mathfrak{B}^{(1)} = \begin{pmatrix} B_1^{(1)} \\ B_3^{(1)} \\ \vdots \\ B_M^{(1)} \end{pmatrix} \\ \mathfrak{B}^{(2)} = \begin{pmatrix} B_1^{(1)} + W_{2,1}^{(2)}\sigma\left(B_2^{(1)}\right) \\ B_3^{(1)} + W_{2,2}^{(2)}\sigma\left(B_2^{(1)}\right) \\ \vdots \\ B_K^{(1)} + W_{2,K}^{(2)}\sigma\left(B_2^{(1)}\right) \end{pmatrix} \end{cases} \quad (4)$$

Nous avons donc la possibilité de modifier l'architecture du réseau à l'échelle des neurones en ne travaillant que sur les poids W. Cependant, cette forme d'élagage, bien que simple à exploiter, est très contrainte en termes de nombre d'opérations retirées. En relâchant la contrainte de la structure (élagage non-structuré) des opérations retirées il est possible de retirer beaucoup plus d'opérations. Ce genre de méthode repose sur des implémentations efficaces des calculs en matrices creuses. Dans ce cas l'architecture n'est pas modifiée à proprement parler.

Importance ou Ressemblance Dans ce qui précède, nous avons supposé connaître la ligne à supprimer dans la matrice $W^{(1)}$. En pratique c'est cette connaissance qui caractérise la difficulté de l'élagage de réseau de neurones. Autrement dit, comment savoir quel neurones ou opérations peuvent être supprimés sans diminuer significativement la précision du modèle ? Il existe alors deux grands paradigmes dans la littérature. Le premier est la mesure de l'importance : il s'agit de déterminer les neurones les moins importants dans la décision du modèle et de les éliminer. La seconde est la mesure de la similarité et consiste à combiner les neurones similaires. Commençons par la mesure de l'importance. La procédure naïve et qui sert encore souvent de référence consiste à simplement mesurer la norme L^p des poids correspondant à chaque norme pour obtenir un vecteur I de normes :

$$I = \begin{pmatrix} \left\|\left(W_{1,1}^{(1)} \cdots W_{1,N}^{(1)}\right)\right\|_p \\ \vdots \\ \left\|\left(W_{M,1}^{(1)} \cdots W_{M,N}^{(1)}\right)\right\|_p \end{pmatrix} \quad (5)$$

La ligne à élaguer est alors simplement l'indice du minimum du vecteur d'importance I. Dans la littérature, il existe toutefois des critères plus élaborés [7, 8, 9, 10]

Dans le cas de l'élagage par similarité, nous allons grouper les neurones par ressemblance. Une façon de

faire consiste à simplement mesurer une matrice de distance D entre les neurones. Par exemple,

$$D = \begin{pmatrix} \left\|(W_{1,1}^{(1)} - W_{1,1}^{(1)} \cdots W_{1,N}^{(1)} - W_{1,N}^{(1)})\right\|_p & \cdots & \left\|(W_{1,1}^{(1)} - W_{M,1}^{(1)} \cdots W_{1,N}^{(1)} - W_{M,N}^{(1)})\right\|_p \\ \vdots & \ddots & \vdots \\ \left\|(W_{M,1}^{(1)} - W_{1,1}^{(1)} \cdots W_{M,N}^{(1)} - W_{1,N}^{(1)})\right\|_p & \cdots & \left\|(W_{M,1}^{(1)} - W_{M,1}^{(1)} \cdots W_{M,N}^{(1)} - W_{M,N}^{(1)})\right\|_p \end{pmatrix} \quad (6)$$

Supposons qu'il s'agisse des neurones correspondant aux lignes 1 et 2. Nous allons définir un nouveau neurone fusion qui aura pour poids associé la moyenne des poids des deux neurones fusionnés. On a alors

$$\begin{cases} \mathfrak{F}(X) = \mathfrak{W}^{(2)}\sigma\left(\mathfrak{W}^{(1)}X + \mathfrak{B}^{(1)}\right) + \mathfrak{B}^{(2)} \\ \mathfrak{W}^{(1)} = \begin{pmatrix} \frac{W_{1,1}^{(1)}+W_{2,1}^{(1)}}{2} & \cdots & \frac{W_{1,N}^{(1)}+W_{2,N}^{(1)}}{2} \\ W_{3,1}^{(1)} & \cdots & W_{3,N}^{(1)} \\ \vdots & \ddots & \vdots \\ W_{M,1}^{(1)} & \cdots & W_{M,N}^{(1)} \end{pmatrix} \\ \mathfrak{W}^{(2)} = \begin{pmatrix} W_{1,1}^{(2)} + W_{1,2}^{(2)} & W_{1,3}^{(2)} & \cdots & W_{1,M}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{K,1}^{(2)} + W_{K,2}^{(2)} & W_{K,3}^{(2)} & \cdots & W_{K,M}^{(2)} \end{pmatrix} \\ \mathfrak{B}^{(1)} = \begin{pmatrix} \frac{B_1^{(1)}+B_2^{(1)}}{2} \\ B_3^{(1)} \\ \vdots \\ B_M^{(1)} \end{pmatrix} \\ \mathfrak{B}^{(2)} = \begin{pmatrix} B_1^{(2)} \\ B_3^{(2)} \\ \vdots \\ B_K^{(2)} \end{pmatrix} \end{cases} \quad (7)$$

Nous savons donc comment éditer le réseau de telle sorte à fusionner des neurones similaires. Pour aller plus loin, il existe de nombreux travaux comme [11, 12].

Évaluations Empiriques Pour l'évaluation des performances de l'élagage, la base de données standard est Cifar10. A noter toutefois qu'elle n'offre pas un test réaliste. D'autres bases de classification plus complexes comme ImageNet ainsi que d'autres tâches, sont de plus en plus souvent abordées.

L'architecture standard est ResNet mais également sa variante Wide-ResNet. Depuis peu, on observer des évaluations sur MobileNet. Cependant, sur ce dernier, l'essentiel des paramètres se trouvant dans la tête de prédiction, on mesure le nombre de paramètres retirés parmi les paramètres en dehors de la tête de prédiction.

En matière de métrique, il y a une double mesure de la précision par rapport au nombre de paramètres retirés. Il est courant de rapporter la réduction du nombre de FLOPs (floating points operations) en plus ou place du nombre de paramètres.

Nous vous proposons les résultats suivants : le pourcentage de paramètres retirés selon le réseau de neurone et la tache permettant de préserver la précision du modèle. Ces valeurs sont listées dans le tableau 1. Les points essentiels sont au nombre de deux. Premièrement, les résultats sur Cifar10 ne sont pas de bons indicateurs des performances sur ImageNet (ou sur des cas réels, en général). Deuxièmement, il est difficile d'obtenir des accélérations de l'ordre de ×3 avec de l'élagage comme c'est le cas avec la quantification, comme nous le verrons dans la suite de cet article.

modèle	tache	paramètres retirés
ResNet 56	Cifar 10	≈90%
Wide ResNet 28-10	Cifar 10	≈75%
ResNet 50	ImageNet	≈60%
MobileNet V2	ImageNet	≈40%

Table 1 – Performances en élagage, on rapporte le pourcentage de paramètres retirés qui ne dégrade pas la précision du modèle

IV/ Décompositions et recompositions

Dans la section précédente, nous avons présenté des solutions permettant d'accélérer les calculs en réduisant le nombre d'opérations à réaliser. Cela s'est traduit notamment par la suppression d'éléments ou de lignes dans la matrice des poids d'une couche. Nous allons maintenant travailler à l'échelle inter-couches des réseaux en présentant deux méthodes : la décomposition de couches (*tensor decomposition*) et la recomposition de couches (*folding*).

Décomposition tensorielle La décomposition la plus connue est la décomposition en valeur singulière (SVD).

$$W^{(1)} = U^{(1)}D^{(1)}V^{(1)} \quad (8)$$

où U et V sont des matrices unitaires et D est une matrice diagonale. Dans le cas où le rang r de $W^{(1)} \in \mathbb{R}^{M\times N}$ satisfait $r < \frac{MN}{M+N}$ alors la décomposition $U \times (DV \times X)$ nécessite moins de calculs que la couche d'origine $W^{(1)}X$ tout en effectuant exactement le même calcul (aux imprécisions numériques près). Il existe de nombreuses décompositions plus élaborées dans la littérature [13, 14, 15].

Recomposition de couches Nous avons vu par le prisme de la SVD, une manière d'augmenter la séquentialité d'un réseau tout en réduisant le nombre total d'opération. Cependant, en pratique, ce qui coûte dans l'inférence des réseaux est le va-et-vient des données, autrement dit : la séquentialité. Les couches que l'on aimerait supprimer sont typiquement les couches effectuant de nombreux calculs en séquences. Le parfait exemple de telles couches est celui des couches de normalisation. En effet, cette couche est

définie par $X \mapsto \gamma \frac{X-\mu}{\alpha+\epsilon} + \beta$, requière donc 4 opérations successives.

Heureusement ces couches peuvent être fusionnées dans les couches de convolutions et densément connectées. Ajoutons une telle couche à notre réseau F, afin de montrer comment la fusionner.

$$F: \quad X \mapsto W^{(2)} relu\left(BN\left(W^{(1)}X + B^{(1)}\right)\right) + B^{(2)} \quad (9)$$

Notons ici que l'on remplace la fonction d'activation pour ne pas avoir de situation de notations malheureuse entre la fonction d'activation et la variance de la normalisation.

Explicitement, on effectue le calcul suivant par neurone :

$$F: \quad X \mapsto W^{(2)} relu\left(\gamma \frac{W^{(1)}X + B^{(1)} - \mu}{\sigma+\epsilon} + \beta\right) + B^{(2)} \quad (10)$$

Il est alors possible de réécrire le réseau sans la couche de normalisation pour l'inférence. On obtient alors

$$F: \quad X \mapsto W^{(2)} relu\left(W^{(1)} \frac{\gamma}{\sigma+\epsilon} \times X + \gamma \frac{B^{(1)} - \mu}{\sigma+\epsilon} + \beta\right) + B^{(2)} \quad (11)$$

Ce qui correspond bien à un réseau à deux couches sans normalisation. C'est l'occasion de se rendre compte que la normalisation est uniquement une méthode d'apprentissage et non un élément architectural à proprement parler. Pour aller plus loin nous vous proposons la lecture suivante [16].

Ce processus s'applique également aux couches convolutionnelles et densément connectées lorsque celles-ci ne sont pas séparées par une fonction d'activation. Cela peut se produire dans certains réseaux pour réduire le nombre de paramètres. Dans le cas de deux couches densément connectées, il suffit de remplacer les couches par une couche dont les poids sont le produit matriciel des poids des couches d'origine et dont on peut déduire le biais par un simple développement. Pour le cas de couches convolutionnelles successives, il faut établir les dimensions (k', k') du nouveau noyau 2D :

$$k' = k_1 + (k_2 - 1) \times s_1 \quad et \quad s' = s_1 \times s_2 \quad (12)$$

avec k_1 et k_2 les dimensions respectives des noyaux de la première et deuxième couche à fusionner et s_1 et s_2 leurs strides. Ensuite, les poids W' sont mis-à-jour avec l'algorithme 1

Algorithm 1 Fusion de Convolutions : calcul des poids

Require : $W_1 \in \mathbb{R}^{k_1 \times k_1 \times n_{in} \times n_h}$,
$W_2 \in \mathbb{R}^{k_2 \times k_2 \times n_h \times n_{out}}$ de strides s_1 et s_2

$W' \leftarrow \{0\}^{k' \times k' \times n_{in} \times n_{out}}$
for $a_{in} \in \{1, \dots, n_{in}\}$ **do**
for $a_{out} \in \{1, \dots, n_{out}\}$ **do**
for $x_1, y_1 \in \{1, \dots, k_1\}^2$ **do**
for $x_2, y_2 \in \{1, \dots, k_2\}^2$ **do**
$t \leftarrow \langle W_{2[x_1, y_1, :, n_{in}]}; W_{2[x_2, y_2, :, n_{out}]} \rangle$
$W'_{[s_1 \times x_2 + x_1, s_1 \times y_2 + y_1, a_{in}, a_{out}]} \leftarrow t$
end for
end for
end for
end for
end for

L'algorithme retourne W'

V/ Quantification

Comme précédemment, l'objectif est d'accélérer les calculs coûteux et donc principalement la multiplication matricielle $W X$. Dans les sections précédentes, nous y sommes arrivés en réduisant le nombre d'opérations à effectuer. Dans le cadre de la quantification le gain de performance est obtenu en remplaçant les opérations en arithmétique à virgule flottante sur 32 bits (float32, ou FP pour full-precision), communément utilisées pour l'entraînement des réseaux de neurones artificiels, par des opérations en virgules fixes sur b bits. En général b appartient à {1, 2, 4, 8, 16}.

Il faut donc convertir les éléments qui composent le calcul de f de telle sorte à les faire tenir sur b bits. On appelle Q l'opérateur permettant d'effectuer cette conversion. En conséquence, on obtient

$$Q(X) \in \{-2^{b-1} - 1, \dots, 2^{b-1} - 1\} \; et \; X \in]-\alpha, \alpha[\quad (13)$$

On suppose ici que le support de X est symétrique par simplicité (en général on pose α = max{|X|}). Par exemple, l'opérateur de quantification naïf est défini par

$$Q: X \mapsto \left\lfloor X \times \frac{2^{b-1} - 1}{\alpha} \right\rceil \in \{-2^{b-1} - 1, \dots, 2^{b-1} - 1\} \quad (14)$$

où $\lfloor \cdot \rceil$ désigne l'opération d'arrondi. Dans le cas général, l'équation 1 devient alors

$$f: \begin{matrix} \mathcal{A}^{d_e} \to & \mathcal{B}^{d_S} \\ X \mapsto & \sigma(Q(W)Q(X) + B) \end{matrix} \quad (15)$$

À noter ici que le biais B n'est pas quantifié. Dans la plupart des méthodes, il sera traité séparément par le moteur d'inférence qui est en charge de réaliser les traitements pour une architecture matérielle donnée. Se pose alors un problème : la quantification introduit-elle une erreur conséquente ? En l'état, si le support de X est très différent de $\{-2^{b-1} - 1, \dots, 2^{b-1} - 1\}$ (ce qui est le cas en pratique), la quantification précédente introduit un changement drastique d'échelle. En réalité, il est nécessaire d'introduire une étape de "dé-quantification".

$$Q^{-1}oQ : X \mapsto \frac{\alpha}{2^{b-1}-1}\left\lfloor X \times \frac{2^{b-1}-1}{\alpha}\right\rfloor \in]-\alpha, \alpha[\quad (16)$$

Dans ce cas, on obtient

$$f: \begin{matrix} \mathcal{A}^{d_e} \to & \mathcal{B}^{d_S} \\ X \mapsto & \sigma\left(Q^{-1}\left(Q(W)Q(X)\right) + B\right) \end{matrix} \quad (17)$$

et l'erreur de quantification correspond bien à une erreur d'arrondi à l'échelle des entrées. Explicitons l'équation précédente :

$$f(X) = \sigma\left(\frac{max\{|X|\}}{2^{b-1}-1}\frac{max\{|W|\}}{2^{b-1}-1}\left\lfloor W \times \frac{2^{b-1}-1}{max\{|W|\}}\right\rfloor \times \left\lfloor X \times \frac{2^{b-1}-1}{max\{|X|\}}\right\rfloor + B\right) \quad (18)$$

En observant cette formulation, on constate trois choses. Premièrement, il n'y a aucune raison d'avoir le même nombre de bits utilisés pour quantifier les entrées X et les poids W. Dans la littérature, il est classique de noter W4/A8 une quantification sur 4 bits des poids et sur 8 bits des activations (on appelle activation la sortie d'un neurone après passage par la fonction d'activation. Ainsi, les activations des neurones d'une couche, sont les entrées des neurones de la couche suivante). Deuxièmement, on constate que le facteur d'échelle $\lambda_X = \frac{max\{|X|\}}{2^{b-1}-1}$ appliqué aux entrées est également appliqué aux sorties. En conséquence, il doit être de dimension 1. Cependant, le facteur d'échelle $\lambda_W = \frac{max\{|W|\}}{2^{b-1}-1}$ est appliqué aux poids et aux sorties et peut donc être un vecteur à d_s dimensions. On parle alors de quantification par-canaux (per-channel). Enfin troisièmement, Si *Q(W)* et *Q(X)* tiennent sur *b* bits, leur produit est sujet à l'overflow (modulo). Pour éviter cela, les calculs intermédiaires sont enregistrés sur 32 bits (cela se joue au niveau de la machine). On parle dans ce cas de l'accumulateur en 32 bits.

En pratique, dans les moteurs d'inférence les opérations Q^{-1} et Q sont dissimulées et optimisées afin d'éviter le passage des entiers aux flottants qui aurait un coût non négligeable. Par ailleurs, la quantification offre un second avantage en dehors de la vitesse d'inférence : la réduction de l'espace mémoire. En effet, l'essentiel du coût en mémoire des réseaux reposent sur le stockage des poids. La quantification en plus de diminuer la taille de leur représentation, augmente la redondance et donc l'efficacité de méthode comme le codage de Huffman utilisé dans les méthodes de compression sans perte.

PTQ Supposons que nous avons à disposition un réseau *F* déjà entraîné (comme il en existe de nombreux disponibles gratuitement en open source). L'opérateur précédent *Q* peut s'appliquer directement aux poids de ce réseau. Cependant, ce n'est pas le cas pour les entrées des couches intermédiaires. En effet, en l'absence de données, il nous manque la valeur d'α (i.e. max{|X|}). Pour y remédier, une solution fut proposée par Nagel et al. [17] et consiste à exploiter les statistiques enregistrées dans les couches de normalisation. Dans ce cas, nous avons donc tous les éléments pour faire de la quantification post-entraînement ou PTQ pour Post Training Quantization. Formellement, on note BN une couche de normalisation précédent la couche que nous souhaitons quantifier. Alors

$$BN : X \mapsto \gamma\frac{X-\mu}{\alpha+\epsilon} + \beta \quad (19)$$

En conséquence, si la couche de normalisation est apprise correctement, nous avons $\mathbb{E}[BN(X)] = \beta$ et $\mathbb{V}[BN(X)] = \gamma^2$. On peut donc choisir 6 écart-types (c'est la valeur recommandée par les auteurs mais en pratique n'importe quelle valeur entre 5 et 10 donne des résultats semblables) et avoir BN(X) ∈]β - 6|γ|; β + 6|γ|[. Il suffit alors de propager cette information dans le réseau pour obtenir le support de l'entrée de la couche *f*.

QAT La méthode précédente a plusieurs mérites. En particulier, elle est simple à implémenter et permet d'obtenir une bonne précision en W8/A8 ce qui correspond à une accélération d'un facteur 3 à 4 à l'inférence. Cependant, lorsque les contraintes nécessitent une plus grande compression/accélération du modèle, le PTQ échoue à fournir des modèles précis. Pour cela, il est nécessaire d'utiliser des données. La question devient alors : *peut-on utiliser des méthodes classiques d'apprentissage pour les réseaux quantifiés ?* La réponse rapide est "non mais oui". On parlera alors d'entraînement pour la quantification ou quantization aware training (QAT).

Formellement, les méthodes standards d'apprentissage des réseaux de neurones sont des variantes de la descente de gradient. Or, par la définition de l'opérateur de quantification *Q*, équation 14, ce dernier possède un gradient nul presque partout au sens de la mesure de Lebesgue. Ceci empêche d'appliquer immédiatement les méthodes de descente de gradient. Le problème étant discret, sans a priori, il est NP-difficile. En pratique, les chercheurs ont observé que les méthodes dites par straight through estimator (STE) permettent un apprentissage efficace. Ces méthodes consistent à simplement prétendre que

$$\nabla(Q^{-1}oQ) = Id \quad (20)$$

qui revient à simplement retirer l'opération d'arrondi de l'optimisation. Intuitivement, cela revient à conserver une version à virgule flottante du réseau. Après une première inférence avec le réseau quantifié, on utilise la version à virgule flottante pour rétro-propager l'erreur, avant de se ramener à nouveau à la version quantifiée du réseau pour l'étape suivante du processus d'apprentissage.

Évaluations Empiriques Dans cette section, nous allons discuter des méthodes d'évaluations courantes dans le domaine et des résultats à connaître.

La quantification est un domaine très développé avec des évaluations étendues sur de nombreuses bases de données et différentes architectures de réseau. En pratique, l'essentiel des méthodes sont testées sur la tâche de classification d'ImageNet qui est le test canonique de la vision par ordinateur. Ce test est très important puisque bon nombre de domaines de la vision

par ordinateur réutilisent des modèles pré-entraînés sur ImageNet et partagent une partie non-négligeable de leur architecture. Ce test est en général le plus difficile parmi les plus courants. On notera également Pascal VOC en détection d'objets et en segmentation d'images, MS COCO en détection et CityScapes en segmentation.

Pour ce qui est des architectures, les méthodes sont systématiquement testées sur ResNet 50. Cependant, cette architecture (comme DenseNet et ShuffleNet) est relativement facile à compresser. Depuis quelques années, les modèles compacts représentent le challenge pour la quantification, en particulier MobileNet V2 et V3, ainsi qu'EfficientNet. Par ailleurs, les transformers qui obtiennent une attention croissante en compression ne sont que rarement considérés en quantification à cause du problème suivant : la layer normalization (normalisation utilisée par les transformers).

modèle	W8/A8	W4/A4	Ternaire	Binaire
ResNet 50	≈100%	≈90%	–	–
DenseNet	≈100%	≈90%	–	–
MobileNet v2	≈100%	≈50%	–	–
EfficientNet B0	≈100%	≈50%	–	–

Table 2 – Performances en PTQ sur ImageNet, la précision est donnée en pourcentage de la précision du réseau de départ.

modèle	W8/A8	W4/A4	Ternaire	Binaire
ResNet 50	≈105%	≈100%	≈92%	≈90%
DenseNet	≈105%	≈100%	≈92%	≈90%
MobileNet v2	≈105%	≈90%	≈75%	–
EfficientNet B0	≈105%	≈95%	–	–

Table 3 – Performances en QAT sur ImageNet, la précision est donnée en pourcentage de la précision du réseau de départ.

En effet, cette couche n'utilise pas de statistiques apprises mais est toujours calculée à la volée en flottant. Ceci rend le problème de la quantification très spécifique dans le cas des transformers. En matière de performance, nous vous proposons les tableaux 2 et 3 donnant les chalenges et de l'état de l'art du domaine. Les éléments à retenir sont les suivants :

Premièrement, en PTQ, le ternaire et le binaire restent encore hors de portée des méthodes actuelles.

Deuxièmement, les réseaux compacts sont significativement plus durs à quantifier.

Troisièmement, le QAT permet d'obtenir des résultats parfois supérieurs à ceux du réseau initial en flottant.

Enfin quatrièmement, même en QAT le binaire reste extrêmement difficile pour des réseaux comme MobileNet et EfficientNet.

Subtilités et Astuces Il est important de préciser plusieurs petites astuces en quantification qui jouent un rôle crucial dans la performance des méthodes ! Il est standard de quantifier la première et la dernière couche du réseau en W8/A8 même si le reste est quantifié en binaire ou ternaire. Autrement dit, personne aujourd'hui ne sait faire de réseau précis entièrement binaires pour des tâches réalistes (comme ImageNet). Pour des compressions extrêmes (binaire et ternaire), il est standard de modifier l'architecture de réseau en l'élargissant pour compenser la perte d'information due à la quantification. Tous ces points font de ce domaine un sujet encore ouvert à la recherche ! Voici une liste non-exhaustive de travaux pertinents pour aller plus loin [17, 18, 19, 20, 21, 22].

VI/ Conclusion

Dans cet article nous avons introduit les grands principes de la compression de réseaux de neurones aux travers des trois stratégies les plus courantes : l'élagage, la décomposition et recomposition tensorielles, ainsi que la quantification. Nous avons observé les performances en fonction de l'approche considérée ainsi que de l'information dont nous disposons au moment de la compression du réseau. Ainsi les méthodes qui n'utilisent que peu ou pas de données sont plus respectueuses de la vie privée, en particulier pour des cas d'utilisation sensibles (dans le domaine médical ou militaire par exemple), mais obtiennent des résultats souvent inférieurs aux méthodes qui se permettent un (ré)entraînement du réseau au cours du processus de compression. Nous avons également veillé à fournir au lecteur quelques références vers des méthodes de la littérature plus élaborées et plus performantes afin qu'il puisse approfondir sa connaissance du domaine. Ces références ne sont toutefois pas exhaustives et certaines méthodes, au-delà de la portée de cet article, n'ont pas été abordée, telles que les approches par distillation [23, 24].

VII/ Références

[1] J. Deng, W. Dong, et al. ImageNet : A Large-Scale Hierarchical Image Database. CVPR, 2009.

[2] Mark Everingham, Luc Van Gool, CKI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge 2012 (voc2012). In Results, 2012.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. CVPR, pages 3213–3223, 2016.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CVPR, pages 770–778, 2016.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. CVPR, pages 4510–4520, 2018.

[6] Mingxing Tan and Quoc V Le. Efficientnet : Rethinking model scaling for convolutional neural networks. ICML, pages 6105–6114, 2019.

[7] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. CVPR, pages 11264–11272, 2019.

[8] Guiying Li, Chao Qian, Chunhui Jiang, Xiaofen Lu, and Ke Tang. Optimization based layer-wise magnitude-based pruning for dnn compression. In IJCAI, pages 2383–2389, 2018.

[9] Congcong Liu and Huaming Wu. Channel pruning based on mean gradient for accelerating convolutional neural networks. Signal Processing, 156 :84–91, 2019. 6 12 juillet 2022 Introduction aux méthodes d'accélération de réseaux de neurones Revue 3EI

[10] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Singe : Sparsity via integrated gradients estimation of neuron relevance. arXiv preprint arXiv :2207.04089, 2022.

[11] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Red : Looking for redundancies for data-freestructured compression of deep neural networks. NeurIPS, 34 :20863–20873, 2021.

[12] Edouard Yvinec, Arnaud Dapogny, Kevin Bailly, and Matthieu Cord. Red++ : Data-free pruning of deep neural networks via input splitting and output merging. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.

[13] Davide Bacciu and Danilo P Mandic. Tensor decompositions in deep learning. arXiv preprint arXiv :2002.11835, 2020.

[14] Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavsk`y, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In ECCV, pages 522–539, 2020.

[15] Yinan Wang, Weihong "Grace" Guo, and Xiaowei Yue. Tensor decomposition to compress convolutional layers in deep learning. IISE Transactions, 54(5) :481–495, 2022.

[16] Edouard Yvinec, Arnaud Dapogny, and Kevin Bailly. To fold or not to fold : a necessary and sufficient condition on batch-normalization layers folding. In IJCAI, 2022.

[17] Markus Nagel, Mart van Baalen, et al. Data-free quantization through weight equalization and bias correction. ICCV, pages 1325–1334, 2019.

[18] Guo Cong, Qiu Yuxian, Leng Jingwen, Gao Xiaotian, Zhang Chen, Liu Yunxin, Yang Fan, Zhu Yuhao, and Guo Minyi. Squant : On-the-fly data-free quantization via diagonal hessian approximation. ICLR, 2022.

[19] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different : Recovering neural network quantization error through weight factorization. ICML, pages 4486–4495, 2019. [20] Yichi Zhang, Zhiru Zhang, and Lukasz Lew. Pokebnn : A binary pursuit of lightweight accuracy. In CVPR, pages 12475–12485, 2022.

[21] Edouard Yvinec, Arnaud Dapgony, Matthieu Cord, and Kevin Bailly. Rex : Data-free residual quantization error expansion. arXiv preprint arXiv :2203.14645, 2022.

[22] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Spiq : Data-free per channel static input quantization. arXiv preprint arXiv :2203.14642, 2022.

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. NeurIPS, 2014.

[24] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill : Data-free knowledge transfer via deepinversion. CVPR, pages 8715–8724, 2020.

CONCEPTION ET DEVELOPPEMENT D'UNE VOITURE AUTONOME

AYOUB KARINE, MAHER JRIDI, REMI ADDE ET SEBASTIEN DEMOUSSELLE
ISEN Yncrea Ouest, site de Nantes, L@bISEN, Vision-AD, 33 Quater Chemin du Champ de Manœuvre, 44470 Carquefou, Nantes
ayoub.karine@isen-ouest.yncrea.fr, maher.jridi@isen-ouest.yncrea.fr

Résumé : *L'enseignement par projet présente plusieurs avantages et constitut une forme d'enseignement qui intéresse de plus en plus les élèves d'écoles d'ingénieurs. Ceci est d'autant plus vrai quand il s'agit de projets pluridisciplinaires intégrant des éléments logiciels, matériels et de traitement de données. Le présent projet est le fruit d'un travail réalisé par des étudiants de l'ISEN Yncréa Ouest/Site Nantes. L'objectif était de doter une maquette de voiture de plusieurs fonctionnalités afin qu'elle soit autonome. Plusieurs compétences ont été travaillées à travers ce projet à l'instar de l'électronique numérique, les systèmes embarqués et l'intelligence artificielle et le développement Web. Plus concrètement, plusieurs capteurs ont été déployés dans la voiture afin d'extraire des informations utiles sur l'environnement qui entoure la voiture. Ces informations sont, par la suite, traitées d'une manière directe pour agir sur le moteur et la direction de la voiture soit d'une manière indirecte via des méthodes d'intelligence artificielle implantées sur un GPU, monté sur la voiture, afin que cette dernière réalise des actions de manière autonome. Parmi ces actions, nous citons l'estimation de la vitesse, le stationnement, la détection et la reconnaissance des panneaux de signalisation.*

I/ Introduction

La voiture autonome est un sujet qui faisait déjà énormément rêver il y a 20 ans et qui était alors considéré comme de la science-fiction. Avec l'évolution des technologies, ce rêve s'est petit à petit transformé en réalité et les voitures du marché deviennent de plus en plus « intelligentes ». Plusieurs niveaux d'autonomie ont été définis. Cela va du niveau 0 dans lequel aucune électronique d'assistance n'est employée dans le véhicule jusqu'au niveau 5 qui est considéré comme le stade ultime (peut être utopique) et se matérialise par le fait que le véhicule peut se conduire tout seul, partout et par tous les temps et sans intervention humaine. Aujourd'hui, le consensus industriel et technologique positionne le niveau de maturité au stade 3. Dans ce niveau le système de pilotage est évolué et permet, lors de certaines phases de conduite, de lâcher les mains pour peu que la réglementation l'autorise. Ainsi, certains modèles sont capables de détecter la signalisation (panneaux, lignes, feux tricolores etc.), se stationner de manière presque autonome ou encore adapter leur vitesse au trafic. Ces véhicules peuvent désormais presque se passer de l'action de l'Homme pour la piloter. C'est dans ce contexte que se situe le travail présenté dans cet article qui vise à proposer une voiture autonome en énergie et en déplacement.

II/ Architecture proposée

La voiture autonome proposée est composée de deux parties : une partie matérielle et une partie logicielle.

II.1/ Partie matérielle :

La maquette de la voiture autonome utilisées est : Maverick CRAWLER SCOOT RC. C'est une voiture radiocommandée électrique à quatre roues motrices. Celle-ci peut être commandée grâce à une télécommande radio en 2.4 GHz

Afin de doter la voiture d'une autonomie, plusieurs composants électroniques sont déployés. Ces composants permettent d'acquérir les informations nécessaires à propos de l'environnement extérieur à la voiture. La figure 1 montre les différents composants ainsi que leurs branchements.

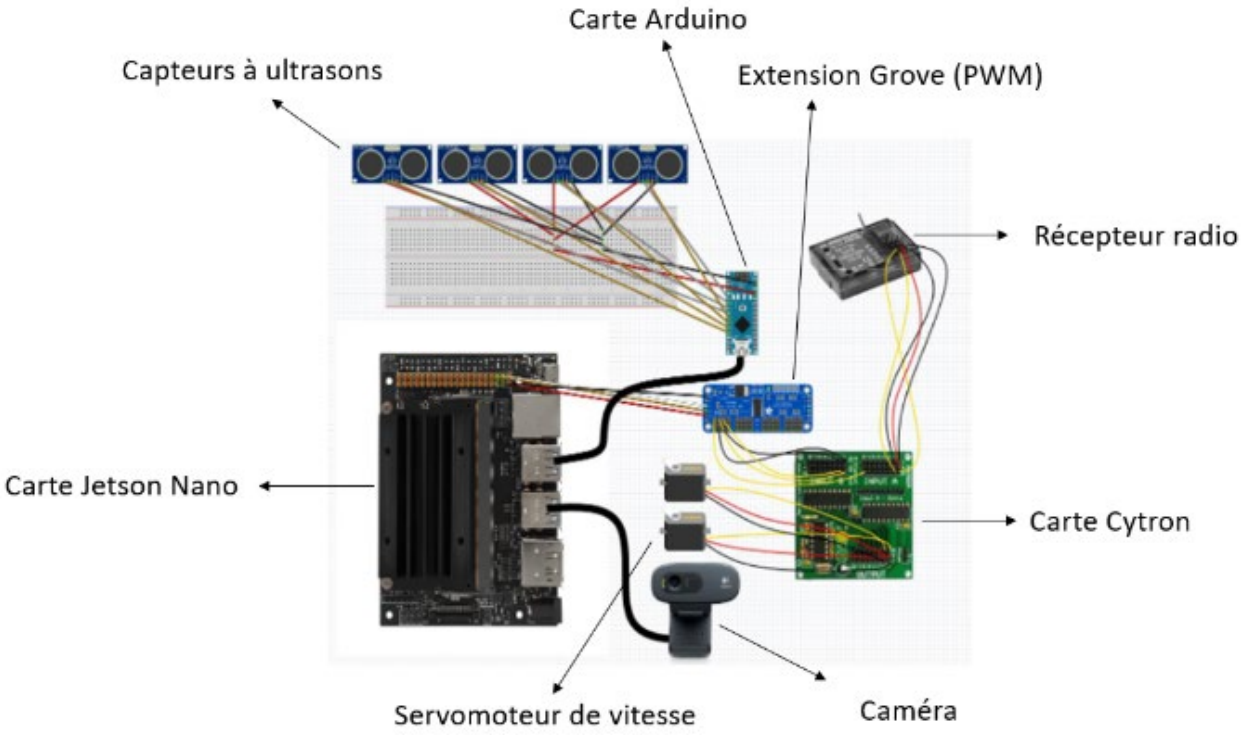


Figure 1 : schéma des branchements de notre système

II.1.1/ Carte Jetson Nano

La brique principale de notre système est la carte Jetson Nano proposée par NVIDIA. Cette carte contient un processeur graphique (GPU, Graphics Processing Unit) lui permettant d'accélérer les calculs, surtout matriciels, d'intelligence artificielle. Elle dispose de 41 broches GPIOs (General Purpose Input/Output).

II.1.2/ Capteurs ultrasons

Des capteurs à ultrasons sont utilisés dans les quatre directions (avant, arrière, gauche, droite) de la voiture afin de calculer les distances. Ces capteurs sont branchés à une carte Arduino Nano reliée à la carte Jetson Nano en série via un câble USB. Le choix de déployer une MCU en plus du GPU n'est pas convenable dans plusieurs applications, d'autant plus que le calcul de la distance à travers le capteur ultrason est très simple et peut se faire dans le GPU. Toutefois, le GPU étant dédié au calcul des réseaux de neurones et les calculs n'étant pas finement parallélisés, le fait de rajouter une composante de calcul supplémentaire nuit fortement à la cadence de traitement vidéo. Pour cette raison, nous étions obligés d'externaliser le calcul de distance par rapport au GPU.

Il faut aussi noter qu'en premier lieu, il a été envisagé de brancher les capteurs à ultrasons directement sur la Jetson Nano. Les capteurs à ultrasons choisis pour le projet sont alimentés en 5V. Cette tension est la même pour le signal renvoyé par l'écho vers la carte. Cependant, les broches des GPIOs de la Jetson Nano supportent au maximum une tension de 3.3V. Il convient donc de réduire la tension de ce signal d'écho. Ceci est possible grâce à un pont diviseur de tension. Pour garder la carte en sécurité, ce pont diviseur de tension réduit la tension de 5V à 2.5V. La figure suivante décrit le schéma de connectique adopté.

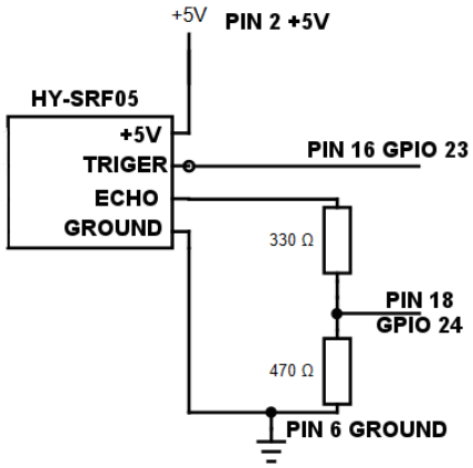


Figure 2. Schéma de liaison du capteur ultrason au GPU

II.1.3/ Extension Grove (PWM)

Une extension PWM (Pulse Width Modulation) a aussi été ajoutée et reliée en I2C (Inter Integrated Circuit Bus) à la Jetson Nano. Celle-ci propose 16 signaux PWM pour seulement trois proposés par la Jetson Nano. Le premier est utilisé pour changer le mode de fonctionnement de la voiture (manuel et autonome). Le second commande le variateur du moteur. Le dernier vise à commander le servomoteur de direction.

II.1.4/ Multiplexeur : la carte cytron

Pour le changement de mode (manuel/autonome), un multiplexeur à deux entrées et une sortie est exploitée. La première entrée est reliée au récepteur radio de la maquette et la seconde entrée est reliée aux PWM de l'extension gérant la direction et la vitesse. Le signal du changement de mode est lui relié à la broche n°8 de l'entrée A (c'est ce port qui gère le changement de mode).

II.1.5/ Caméra

Une webcam USB est reliée à la carte Jetson Nano afin d'acquérir les images des panneaux de signalisation placés devant la voiture. La carte Jetson Nano est aussi reliée à internet grâce à une clé WiFi branchée en USB. Les images acquises ont une résolution HD 1920x1080 largement suffisante pour l'application. Le lien Wifi peut être utilisé pour des procédures de débogages et pour faire du dashbording sur l'historique des opérations.

II.1.6/ Batterie

Un autre composant non-représenté sur le schéma de la figure 1 à savoir la batterie est utilisé. En effet, la maquette est autonome en énergie. La Jetson Nano est donc alimentée par une batterie. Cependant, le microcontrôleur doit être alimenté par une tension de 5V (Volts) et les batteries (en charge complète) délivrent une tension à vide de 8.4V. Un régulateur de tension réglable était alors utilisé, permettant à la Jetson Nano d'être alimentée sur une plage de batterie allant de 8.4V à 7.3V (dans cette plage de tension, le régulateur fournit bien une tension de 5V).

La figure 3 illustre le montage des différents composants dans la voiture. Les connectiques sont volontairement mises en valeur sur les schémas pour montrer le nombre de liens entre composants.

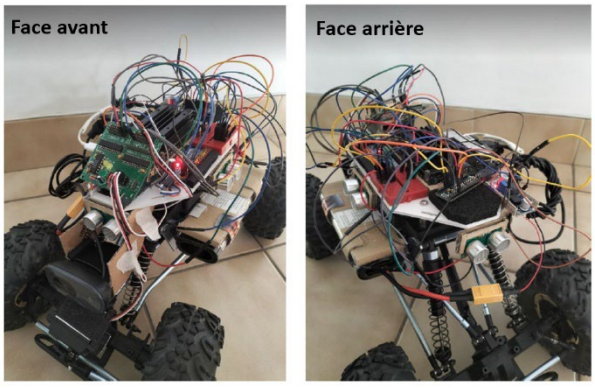


Figure 3 : Voiture autonome développée.

Une fois que tous les éléments sont mis bout à bout, des tests ont été réalisés sur l'évaluation des angles et les valeurs des signaux rapports cycliques des signaux PWM. L'ensemble de ces tests est décrit sur la figure 4.

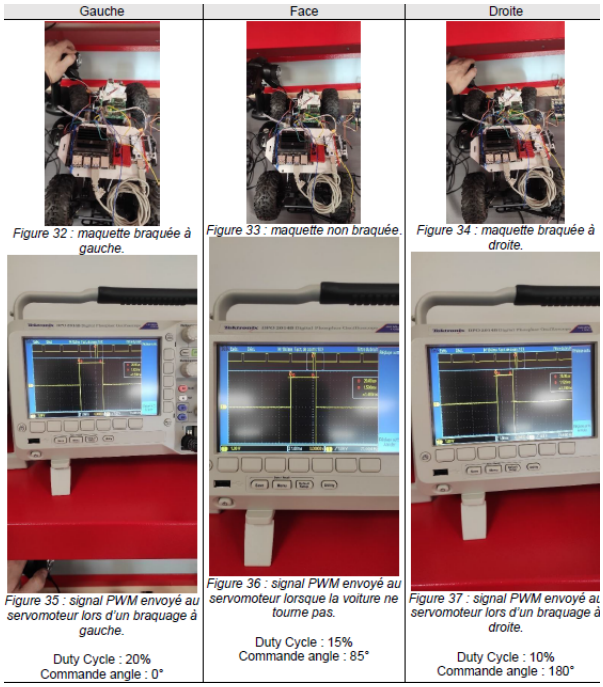


Figure 4 : Evaluation des rapports cycliques des signaux PWM

II.2/ Partie logicielle

Grâce aux composants montés sur la maquette, des fonctionnalités permettant un comportement autonome de la voiture ont été implémentées. Cette autonomie est atteinte via des fonctionnalités comportementales et des techniques d'intelligence artificielle.

Toutes les fonctionnalités ont été développées en langage Python sur la Jetson Nano et en langage C sur la carte Arduino Nano.

II.2.1/ Les fonctionnalités comportementales.

Toutes les fonctionnalités comportementales ont été développées en langage C.

La première fonctionnalité développée est l'estimation de la vitesse. Elle se base sur la vitesse maximale de la voiture (3,6 km/h) et de la consigne PWM pour la commande du au moteur. La vitesse maximale correspond à la consigne PWM maximale, la vitesse en temps réel est estimée par une règle de proportionnalité.

Nous tenons à préciser que le microcontrôleur (la carte ARDUINO) peut contrôler la direction de la maquette et sa vitesse grâce aux signaux PWM. Ceux-ci sont commandés grâce à une librairie appelée Servokit. Cette dernière permet d'envoyer des consignes sous forme d'angles compris entre 0 et 180 degrés.

La voiture peut, de plus, adapter sa vitesse en fonction aux obstacles qui lui font face à l'aide des capteurs à ultrasons. Au moment où un objet est détecté à une distance inférieure à 50 cm, cette même distance est enregistrée et un ratio est effectué avec les nouvelles distances mesurées à chaque instant. Ceci permet de réduire la consigne de vitesse (entre 0 et 180) en fonction de la distance. Évidemment, si la distance augmente, la

consigne augmente elle aussi de nouveau jusqu'à revenir à sa valeur initiale.

Grâce aux capteurs à ultrason droit et arrière, la maquette peut se garer de manière autonome. Pour ce faire, le premier ultrason droit cherche une place assez grande sur laquelle se garer et le deuxième ultrason permet à la maquette de s'arrêter en cas de présence d'un obstacle. Si cet obstacle disparaît, la voiture reprend sa manœuvre jusqu'à l'arrêt final.

Finalement, la caméra déployée dans la maquette permet de filmer en temps réel la scène en face de la voiture. Les vidéos acquises vont permettre à la voiture de reconnaître les panneaux de signalisation afin de les respecter de manière autonome via des méthodes d'intelligence artificielle.

II.2.2/ Détection et reconnaissance des panneaux de signalisation via l'intelligence artificielle.

Pour que la voiture respecte les panneaux de signalisation de manière autonome, nous avons contribué à embarquer des méthodes d'intelligence artificielle (IA) dans la carte Jetson Nano. L'objectif est que la voiture détecte et reconnaisse un panneau de signalisation et réagisse en conséquence. De ce fait, deux algorithmes doivent être développés. Un premier pour la détection du panneau et un deuxième pour sa classification (reconnaître le signe du panneau).

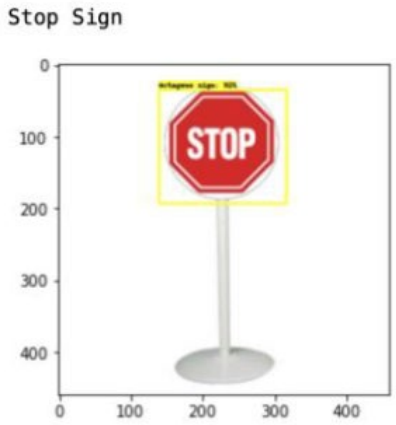


Figure 5 : Détection et reconnaissance du panneau de signalisation stop

L'intelligence artificielle regroupe les méthodes permettant d'imiter et de simuler l'intelligence humaine. Parmi ses disciplines, citons l'apprentissage automatique qui vise à doter les machines de la capacité d'apprendre à partir de données. Nous distinguons deux familles de méthodes d'apprentissage automatique. La première famille concerne les méthodes supervisées qui se base sur les données étiquetées (données + vérités terrains) pour construire un modèle de décision. Quant à la deuxième famille, à savoir l'apprentissage non-supervisé, les données utilisées sont soit non-étiquetées soit étiquetées mais les étiquettes ne sont pas utilisées pour construire le modèle de décision. De manière générale, si les étiquettes ont des valeurs continues on parle de la régression (Exemple : prédire le prix d'une voiture en se basant sur l'historique des prix et tous les paramètres qui influent sur le prix) et si elles ont des valeurs discrètes, il s'agira d'un problème de classification (Exemple : prédire si l'image contient un chat ou pas).

Pour les méthodes supervisées, les données (dans notre cas les images) doivent être divisées en trois parties. La première partie sert à faire construire le modèle d'apprentissage en apprenant sur les images d'entrée. Elle est appelée base d'apprentissage. Ensuite, la base de validation est utilisée pour trouver les meilleurs paramètres du modèle d'apprentissage. Enfin, la base de test permet de tester le modèle d'apprentissage en calculant différentes mesures d'évaluation à l'instar du taux de bonne classification. Cette mesure est obtenue en divisant le nombre de données bien classifiées de la base de test sur le nombre total de données de test. C'est cette famille de méthodes, à savoir l'apprentissage supervisé, qui a été exploitée dans le présent projet. Les étiquettes sont représentées, dans notre étude, par des classes. Plus précisément, nous avons fait recours à 7 classes (chaque panneau correspond à une classe) représentées sur la figure 6.

Figure 6 : Panneaux de signalisation utilisés.

Plusieurs méthodes d'apprentissage supervisé existent dans la littérature. Parmi celles-ci, les réseaux de neurones ont montré leur suprématie dans plusieurs domaines : traitement automatique du langage, vision par ordinateur, …. Un réseau de neurones est composé de plusieurs couches. Chaque couche contient différents neurones contenant des valeurs numériques. Entre deux couches, des opérations mathématiques sont effectuées en utilisant les paramètres du réseau à savoir les poids et les biais. Avec la disponibilité de grandes bases de données et le développement de machines dotées d'une grande capacité de calculs, la profondeur (le nombre de couches) des réseaux de neurones a été nettement augmentée donnant naissance à une nouvelle branche de l'intelligence artificielle à savoir l'apprentissage profond.

Afin de traiter les images/vidéos, un modèle d'apprentissage profond nommé réseau de neurones à convolution [1] est souvent exploité. Ce type de modèle se basent sur des convolutions consécutives entre des filtres de petites tailles et les différentes parties du contenu 2D. Les résultats de ces convolutions permettent de générer des cartes caractéristiques qui modélisent des détails visuels de l'image. Ainsi,

l'utilisation de plusieurs couches de convolution permet d'acquérir des détails de l'image à différentes échelles.

Pour bien entraîner un réseau de neurones profond, une base de données avec un grand nombre de données est nécessaire. Néanmoins, une technique de transfert d'apprentissage permet à ce type de données de traiter les bases de données de petites tailles. Cette approche consiste à utiliser des modèles pré-entrainé sur d'autres grandes bases de données pour réapprendre d'autres modèles sur de nouvelles données. Plus précisément, les paramètres du réseau (biais et poids) sont initialisés avec les valeurs d'un réseau pré-entraîné sur une grande base de données à l'instar d'ImageNet qui est composée de plus d'un million d'images réparties en 1000 classes. Nous avons adopté, dans le présent travail, cette technique sur une base de données constituées de 17 345 images.

Afin de manipuler les réseaux de neurones à convolution, nous nous sommes basés sur le langage Python et les bibliothèques Tensorflow [2] et Keras [3].

Pour la détection, nous avons utilisé un modèle déjà pré-entrainé à savoir SSD (Single-Shot Detector) [4]. Plus concrètement, la détection permet de dessiner un rectangle autour de l'objet détecté. Cette zone d'intérêt détecté est ensuite classifiée pour atteindre l'objectif de la reconnaissance du panneau via la méthode MobileNet [5]. Cette méthode a une faible empreinte mémoire et est très utilisée dans les applications temps réel. Le taux de classification global obtenu est de 96%.

Les modèles de l'IA correspondant à la détection et à la reconnaissance des panneaux de signalisation sont ensuite embarqués dans la carte Jetson Nano montée dans la voiture. Pour ce faire, la bibliothèque Tensorflow Lite [6] a été utilisée. Ainsi, chaque « frame » acquis par la caméra est traité en temps réel par la méthode d'IA. Ensuite, la voiture agit en appliquant la consigne du panneau. Par exemple, la voiture s'arrêtera devant un panneau STOP.

III/ Expérimentations

III.1/ Gestion de la vitesse

Le taux de classification étant à 96%. Cette partie permet de connaitre le panneau de signalisation et d'adapter par la suite la trajectoire. Pour la phase d'expérimentation, les objectifs pour la maquette sont donc les suivants :

- Respect des limitations de vitesse (la maquette va au maximum de la vitesse autorisée si c'est possible).
- Dans le cas où la maquette arrive trop vite sur un véhicule en face d'elle, elle adapte sa vitesse pour le suivre à une distance de sécurité ou s'arrête s'il est arrêté.

La détection d'un obstacle se fait en fonction de la distance que mesure le capteur à ultrasons. L'expression de cette distance se fait grâce au temps écoulé et à la vitesse du son. Le tout divisé par deux car l'onde fait un aller-retour. La formule ci-dessous exprime la distance :

$$Distance = tempsécoulé * vson2 * 100 \text{ (cm)}$$

Cependant, cette distance est celle comprise entre le capteur et l'objet, mais ce capteur n'est pas forcément au bord de la maquette. Il faut donc prendre en compte la distance entre l'avant de la maquette et la position du capteur.

La distance avec l'objet face à la maquette est donc la suivante :

$$Distanceobjet = tempsécoulé * vson2 * 100 - Distancecapteur \text{ (cm)}$$

Ce même principe est appliqué à tous les capteurs à ultrasons de la maquette. Ensuite, en connaissant la distance entre la maquette et l'objet ainsi que la vitesse de la voiture, il est possible de déterminer les éléments suivants :

- Est-il possible d'éviter un impact ?
- Quelle est la consigne permettant de réduire la vitesse de la maquette et éviter l'impact ?

Une distance critique (de sécurité) d'un obstacle à laquelle la maquette doit s'arrêter (ou suivre l'objet s'il bouge) a été déterminée : $Distancecritique = 15cm.$

La formule de la distance réelle (appelée distance à parcourir) dont dispose la maquette pour s'arrêter est la suivante :

$$Distanceàparcourir = Distanceobjet - Distancemin.$$

Les conditions qui ont été définies pour déterminer le comportement de la maquette sont les suivantes (la distance des trois points ci-dessous est la distance à l'objet sans la distance critique).

- Distance de l'objet détecté > 40cm : aucune nécessité de s'arrêter.
- Distance de l'objet détecté ≤ 40cm : freinage jusqu'à l'arrêt total (si objet fixe), sinon suivi de cet objet à une vitesse adaptée.
- Distance de l'objet détecté ≤ Distance critique : arrêt d'urgence avec les gaz à fond en arrière (ou à 0).

Comme dit précédemment, la maquette est censée se déplacer au maximum de la vitesse autorisée (ou en premier lieu, à la vitesse définie au début du programme). Cette vitesse s'appelle la vitesse visée (dont la formule est la même que la vitesse estimée) :

$$Vvisée = Vmax * ConsigneVvisée$$

La consigne que l'on donne à la voiture est définit par la formule suivante : $ConsigneVvisée = VviséeVmax$ (%)

Cette consigne est modifiée lorsqu'un obstacle est détecté à 40 cm ou moins. Elle diminue si l'obstacle ne bouge pas et se stabilise ou augmente à nouveau si l'obstacle accélère (comme une voiture). Enfin, si la vitesse de l'objet détecté vient à dépasser celle de la maquette, la consigne finit par ne plus être modifiée car la distance à l'objet en question est supérieure à 40 cm. Donc, dans une période d'adaptation de vitesse, la formule définissant la consigne est la suivante :

$$Consigne = ConsigneVvisée * DistanceObjetDistanceObjetInit$$

La figure 7 montre l'adaptation

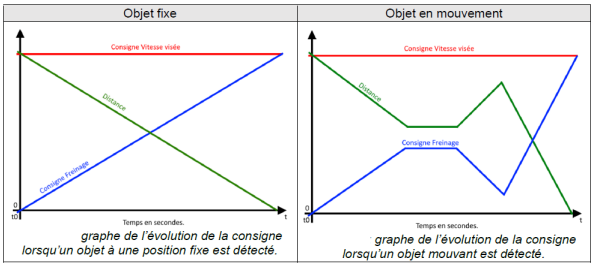


Figure 7 : évolution de la consigne en fonction de la détection de l'objet

III.2/ IHM Web

Pour améliorer l'expérience utilisateur, une interface web a été développée (cf. figure 8). Cette interface représente le tableau de bord du système développé. Elle communique avec la partie matérielle de la voiture via les websockets. Elle permet :

- de visionner en temps réel ce que voit la voiture avec la webcam embarquée,
- d'afficher la vitesse de la voiture,
- de contrôler le mode de la voiture (autonome/manuel),
- d'activer ou de désactiver le mode parking.

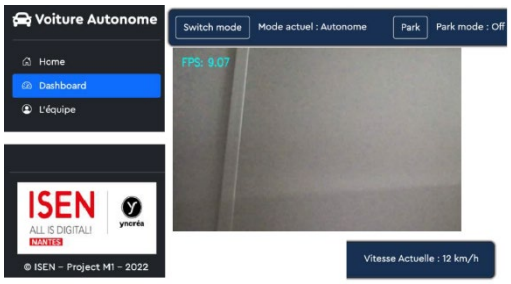


Figure 8 : Tableau de bord de notre interface web.

IV/ Conclusion

Ce projet, dont l'idée directrice était de pouvoir rendre une maquette de voiture télécommandée autonome sans intervention de l'être humain a abouti sur des résultats intéressants. Les technologies utilisées, telles que la Jetson Nano ont permis de pouvoir utiliser une intelligence artificielle embarquée puissante qui peut reconnaître des panneaux de signalisation avec une erreur assez faible et des performances acceptables. De plus, les différents capteurs à ultrasons permettent à cette maquette de connaître son environnement et éviter tout problème de collision. Enfin, cette maquette a la capacité de pouvoir se garer de façon autonome en créneau à droite lorsque la taille de la place détectée est suffisante.

V/ Bibliographie

[1] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.

[2] https://www.tensorflow.org/

[3] https://keras.io/

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu: "SSD: Single Shot MultiBox Detector", 2016

[5] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[6] https://www.tensorflow.org/lite?hl=fr

CONCEPTION, REALISATION, ET ETALONNAGE D'UNE CELLULE D'EFFORTS 6 AXES A BAS COUT

YAN BARBINOT [1], FRANÇOIS LOUF [2]
[1] : étudiant en 4ème année de l'ENS Paris-Saclay
[2] : Directeur du DER de Génie Mécanique de l'ENS Paris-Saclay

Résumé : *Les capteurs d'efforts mono-axes sont présents dans de nombreuses applications industrielles ou à vocation pédagogique. Leur coût est désormais très abordable. Cependant, lorsque l'on souhaite obtenir une mesure de chargements complexes (3 axes, ou 6 axes), les produits se font plus rares et sont souvent très onéreux. L'objectif de cet article est de présenter la conception, la réalisation et un exemple d'utilisation d'une cellule 6 axes à faible coût utilisable par exemple dans le cadre de projets pédagogiques*

Mots clés *: cellule d'effort, arduino, cellule multi-axe, impression 3D*

I/ Introduction

Dans le domaine industriel, les capteurs d'efforts mono-axe sont couramment utilisés pour mesurer les actions mécaniques transmises entre deux solides selon une seule direction. Il en existe de plusieurs types, mesurant une résultante ou un couple, qui dépendent notamment du niveau de charge à supporter et de la précision de mesure souhaitée. Par exemple, le capteur de pesée à jauges de déformations est fréquemment implanté dans les systèmes mécaniques pour mesurer des efforts uni-axiaux de traction ou compression. Son faible coût, sa taille variable et sa robustesse de mesures en font un capteur de choix pour les systèmes industriels et didactiques.

Dans un mécanisme, pour mesurer un effort exercé par une pièce S_1 sur une autre S_2, l'idée est de créer une liaison glissière entre les deux solides puis d'insérer un capteur d'effort mono-axe en parallèle afin de bloquer le degré de liberté correspondant. Ce principe peut être illustré à l'aide du schéma proposé en Figure 1.

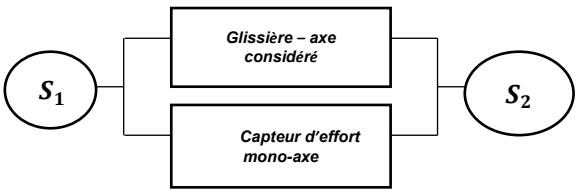


Figure 1 - Association de base d'un capteur d'effort mono-axe

Ce schéma peut être adapté dans le cas où l'on veut mesurer un moment exercé par une pièce S_1 sur une autre S_2 comme indiqué en Figure 2. Cette représentation nous aidera à définir le concept de capteur multi-axes.

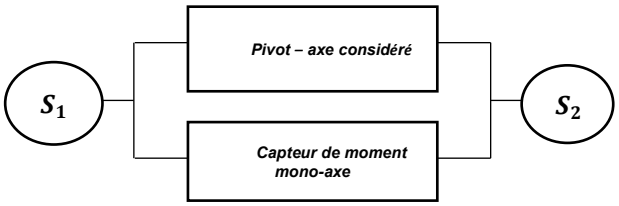


Figure 2 - Association de base d'un capteur de moment mono-axe

Pour concevoir des capteurs multi-axes, l'idée la plus simple est de réaliser une association en série des blocs élémentaires présentés en

Figure 1 et Figure 2. A titre d'exemple, le concept d'un capteur tri-axe mesurant les efforts dans trois directions, peut être représenté par le schéma de principe en Figure 3.

Il est facile de se représenter l'idée d'un capteur avec une association en série en faisant un parallèle avec la robotique. Un bras de robot est souvent une association en série de liaison élémentaires motorisées de type pivot et glissière. Un capteur avec une structure en série suit le même principe. Là où les liaisons sont motorisées pour un robot en série, elles sont bloquées par le capteur mono-axe correspondant au degré de liberté considéré dans un capteur multi-axes.

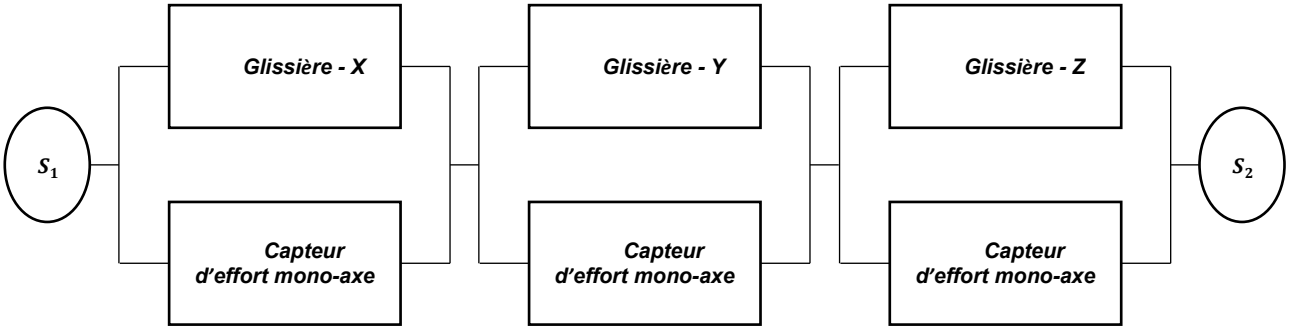


Figure 3 - Exemple d'association en série pour un capteur d'efforts tri-axes

Lorsque les chargements se complexifient – au-delà de trois composantes à caractériser – ou que les actions mécaniques à mesurer sont méconnues, des capteurs d'efforts six axes sont requis. Ils permettent de caractériser complètement les composantes d'un torseur d'actions mécaniques s'exerçant sur le corps d'épreuve à savoir trois couples et trois résultantes selon trois directions de l'espace. Il existe plusieurs tailles et géométries différentes suivant le type d'application, le coût étant évidemment fonction de ces paramètres.

Pour ce type de capteur, il est facile de se représenter une structure en série comportant trois liaisons glissières et trois liaisons pivots. De plus, comme expliqué ci-avant, cette solution donne directement accès à chaque quantité d'intérêt dans la mesure où les capteurs mono-axes de la chaine en série sont indépendants les uns des autres, les directions des efforts/moments mesurés étant orthogonales entre elles. Cette solution, bien que simple à concevoir, possède néanmoins quelques inconvénients. Tout d'abord, il faut pouvoir assurer une certaine rigidité au montage, ce qui contraint la conception des liaisons élémentaires. Ensuite, comme les liaisons sont en série, les erreurs de mesures causées par les jeux et les frottements s'ajoutent au fur et à mesure que l'on remonte la chaine. Enfin, dans certains systèmes, il faut pouvoir minimiser l'encombrement spatial, ce qui n'est pas chose aisée avec une association en série.

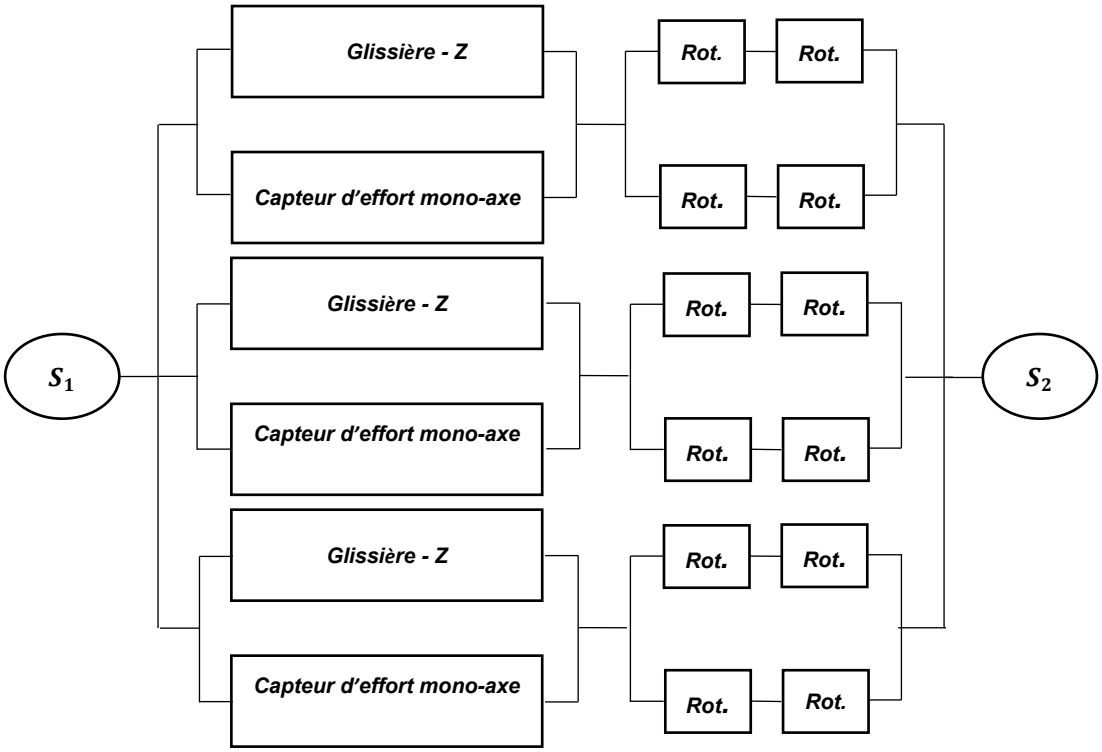


Figure 4 – Exemple d'association de capteurs en parallèle pour réaliser un capteur d'efforts tri-axes basé sur une architecture de type « Delta » ; le terme « Rot » désigne une liaison rotule ; chaque binôme de rotule en série permet de réaliser une bielle, chaque binôme de bielle en parallèle permet de réaliser un parallélogramme déformable

Une autre solution consiste alors à imaginer des structures de capteurs avec des liaisons disposées en parallèle plutôt qu'en série. De nombreux capteurs trois axes et six axes utilisent ce principe. Par exemple, pour mesurer trois résultantes selon les trois axes de l'espace entre deux solides, il est possible d'utiliser une structure Delta. En reprenant notre formalisme, le principe du capteur peut être explicité sur la Figure 4.

Dans cette configuration, les degrés de liberté ne sont plus indépendants : un chargement exercé par $\boldsymbol{S_1}$ sur $\boldsymbol{S_2}$ sera mesuré par l'ensemble des capteurs mono-axes et la valeur de la mesure sera une combinaison linéaire des valeurs retournées par chacun des capteurs mono-axes. L'association en parallèle a l'avantage de posséder une meilleure rigidité par rapport à une association en série. L'encombrement spatial est également moindre mais l'étalonnage plus difficile car les degrés de libertés sont liés.

Suite à ce développement, il est légitime de préciser un point sur la fabrication de ces liaisons élémentaires au sein d'un capteur. Il est impossible d'utiliser des guidages mécaniques standards dans ce type de capteur car cela engendrerait beaucoup de jeux et de frottements qui nuiraient à la qualité des mesures. Ainsi, quelle que soit la solution retenue – association en série ou association en parallèle – les constructeurs utilisent généralement des liaisons souples.

Dans l'industrie, le coût des capteurs d'efforts 6 axes varie selon les dimensions, les capacités de charge, et la précision de mesure notamment, et peut atteindre rapidement plusieurs milliers d'euros. Dans le cadre d'applications pédagogiques, on peut avoir besoin de cellules de ce type, mais on peut également accepter une précision moindre que dans le cas d'applications industrielles. Le travail présenté dans cet article vise donc à concevoir et réaliser des cellules 6 axes dont le coût ne dépasse pas 150 euros et pouvant être utilisées en enseignement, par exemple, dans le cadre de projets.

Cet article présentera donc les différentes étapes de conception et de fabrication de deux cellules 6 axes exploitant la même structure d'hexapode mais présentant des géométries, des choix de conception, et des performances en termes de niveau de chargement différentes.

- Le modèle A (voir Figure 5) a un encombrement de $\boldsymbol{150mm \times 170mm \times 80\,mm}$ et peut supporter une charge verticale de 70 N environ ;
- Le modèle B (voir Figure 6) a un encombrement de $\boldsymbol{240mm \times 280mm \times 185\,mm}$ et peut supporter une charge verticale de 260 N environ.

Après avoir présenté le concept et proposé un modèle de comportement, la conception et la réalisation de deux cellules 6 axes à faible coût sont présentés. L'instrumentation et la récupération des données mesurées au moyen d'une carte arduino sont également détaillées. Enfin, une application à la mesure des actions mécaniques transmissibles par un drone est présentée.

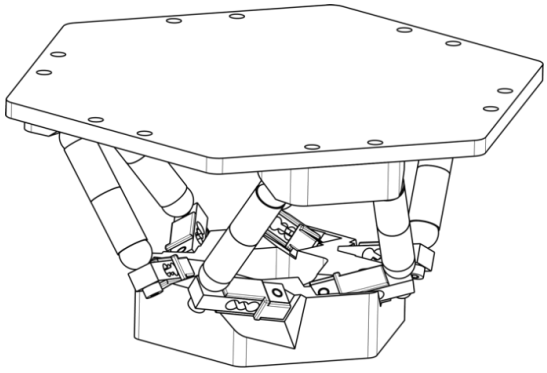


Figure 5 - Vue 3D du modèle A développé

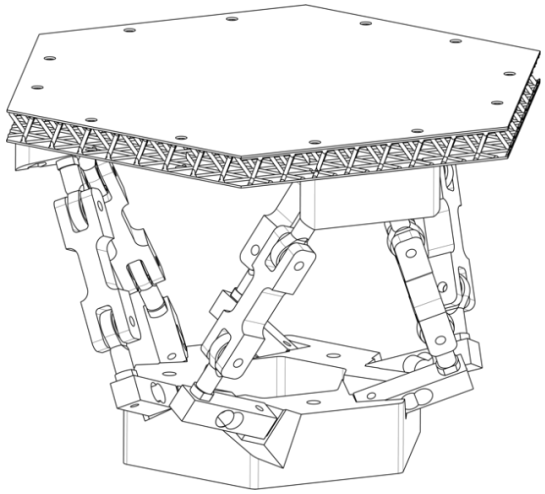


Figure 6 - Vue 3D du modèle B développé

II/ Conception

II.1/ Architecture retenue : hexapode

La géométrie retenue pour les deux capteurs d'efforts six axes à concevoir est celle d'un hexapode. Il s'agit d'une structure parallèle dans laquelle les actions mécaniques imposées au plateau de charge sont transmises à l'embase fixe par l'intermédiaire de 6 bielles bi-rotulées comme montré sur la Figure 7. Chaque bielle est donc sollicitée uniquement en traction-compression et cette sollicitation peut être mesurée par un capteur mono-axe si l'axe de mesure de ce capteur est placé dans l'axe de la bielle.

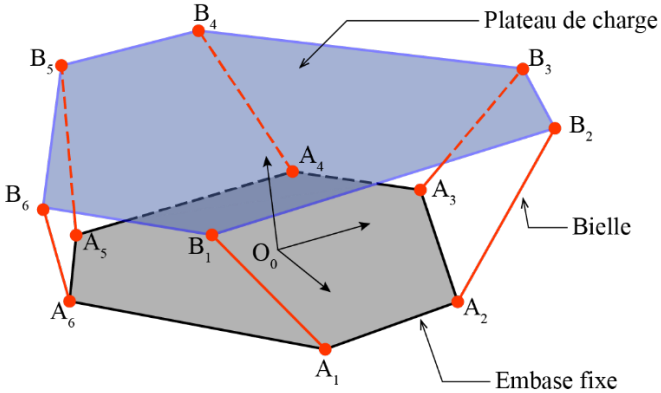


Figure 7 - Modèle filaire représentant l'architecture des cellules 6 axes à concevoir

On rappelle ici que la mesure d'un effort via un capteur mono-axe ne donne pas accès directement à l'une des 6 actions mécaniques appliquées sur le plateau. Il sera nécessaire d'avoir les données des 6 capteurs pour reconstruire la sollicitation imposée au plateau de charge.

Le modèle filaire présenté en Figure 7 permet de réaliser un prédimensionnement de la cellule. Pour cela, les dimensions des pièces sont paramétrées de sorte à pouvoir concevoir des cellules de taille et de capacité de charge adaptées à l'usage. Le prédimensionnement peut s'effectuer soit de manière analytique en considérant un modèle poutre et en réalisant des fermetures géométriques, soit numériquement à l'aide d'un calcul élément finis utilisant des éléments poutres. Ces calculs complexes ne seront pas détaillés ici mais nous invitons le lecteur à aller voir la version détaillée de cet article sur Eduscol (https://eduscol.education.fr/sti/si-ens-paris-saclay).

II.2/ Conception et réalisation

Les choix à réaliser portent sur :

- les liaisons rotules permettant de transmettre les efforts dans l'axe de charge des capteurs d'effort ;
- les bielles ;
- les capteurs d'effort eux-mêmes ;
- le plateau de charge sur lequel seront fixées 6 rotules,
- l'embase fixe sur laquelle seront fixées les 6 capteurs équipés eux même d'une rotule.

Deux types de liaison rotule ont été utilisés :

- Modèle A : des rotules magnétiques dont l'avantage principal est l'absence de jeu de fonctionnement, mais dont le principal inconvénient est la présence d'un frottement sec important dans la liaison ;
- Modèle B : des rotules mécaniques standard dont l'avantage principal est le faible frottement, mais qui peuvent présenter un léger jeu de fonctionnement.

Les bielles réalisées sont adaptées aux rotules retenues. Dans le modèle A, les bielles sont directement constituées des deux parties femelles des rotules assemblées par une vis sans tête. Dans le modèle B, des bielles ont été imprimées en 3D, en ABS.

Les capteurs mono-axe retenus sont des capteurs à jauges de déformation souvent utilisées pour réaliser des systèmes de pesée. Le principe réside en un corps d'épreuve muni de 4 jauges de déformations montées dans un pont de Wheatstone. Sous charge, les résistances des jauges varient, et le déséquilibre du pont est proportionnellement lié à la charge appliquée. La géométrie du capteur, la présence de 6 jauges et leur positionnement permettent de ne mesurer que la composante souhaitée du chargement. Les capteurs retenus sont de deux géométries :

- Modèle A : capteur de capacité 1.5kg, de dimension 6mm x 8mm x 30mm
- Modèle B : capteur de capacité 5.0kg, de dimension 12.7mm x 12.7mm x 55mm

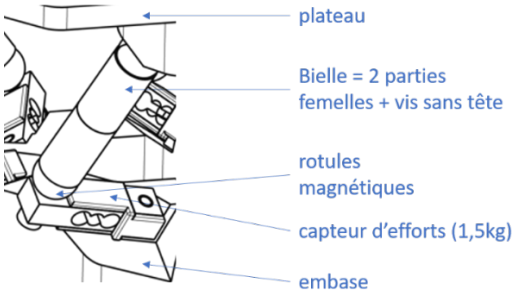


Figure 8 - Réalisation modèle A

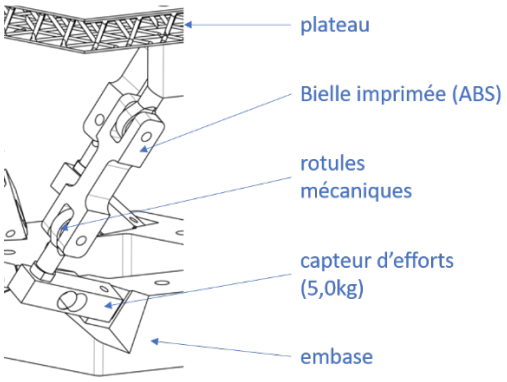


Figure 9 - Réalisation modèle B

Pour les deux modèles, l'embase fixe a été conçue à l'aide d'un modeleur volumique et en exploitant le modèle filaire paramétré comme squelette. Ainsi il est possible d'orienter correctement les plans d'appui des 6 capteurs d'effort et de définir une géométrie 3D de l'embase adaptée. Celle-ci étant de forme complexe, elle est naturellement réalisée en impression 3D ABS.

Pour le modèle A, le plateau supérieur est découpé au jet d'eau dans une plaque de plexiglas tandis que, pour le modèle B, un plateau en composite sandwich a été fabriqué. Les peaux sont des stratifiés carbone-epoxy et l'âme a été réalisée à partir de l'impression 3D d'un treillis à maille tétraédrique. Dans les deux cas, le plateau est suffisamment rigide pour pouvoir transmettre les efforts et assez peu coûteux si l'on dispose des moyens de production associés. Des pièces d'adaptation permettant de fixer les rotules sur le plateau ont été imprimées en ABS pour les deux modèles.

Une fois les solutions techniques choisies, les pièces ont été imprimées 3D et assemblées avec des vis. La réalisation ne nécessite donc pas d'outils haut de gamme ce qui assure l'aspect « low cost ». Une vue 3D des deux modèles conçus et réalisés sont présentés sur les Figure 5 et Figure 6.

II.3/ Mesures des efforts sur chaque axe

Le système d'acquisition est identique pour les deux modèles de cellule et le schéma associé est présenté en Figure 10 : des jauges de déformations sont collées aux corps d'épreuve des capteurs de pesée uni-axiaux et délivrent, par l'intermédiaire de ponts de Wheatstone, des signaux analogiques images du chargement imposé sur le plateau supérieur. Ces signaux sont ensuite amplifiés par des amplificateurs de type HX711 puis traités par un boitier Arduino relié à un ordinateur. Il y a donc au total, pour chaque cellule de charge, six chaînes d'acquisition identiques et indépendantes composées des sous-systèmes énumérés ci-avant.

Une autre solution pour acquérir les signaux de mesure consiste à utiliser un pont d'extensométrie standard à six voies. Les signaux de sorties sont souvent de meilleure qualité mais il est plus difficile de traiter les données car cela peut nécessiter un relevé manuel ou des amplificateurs interfacés un logiciel. Ces ponts d'extensométrie sont donc souvent très coûteux ce qui explique le choix de la première solution.

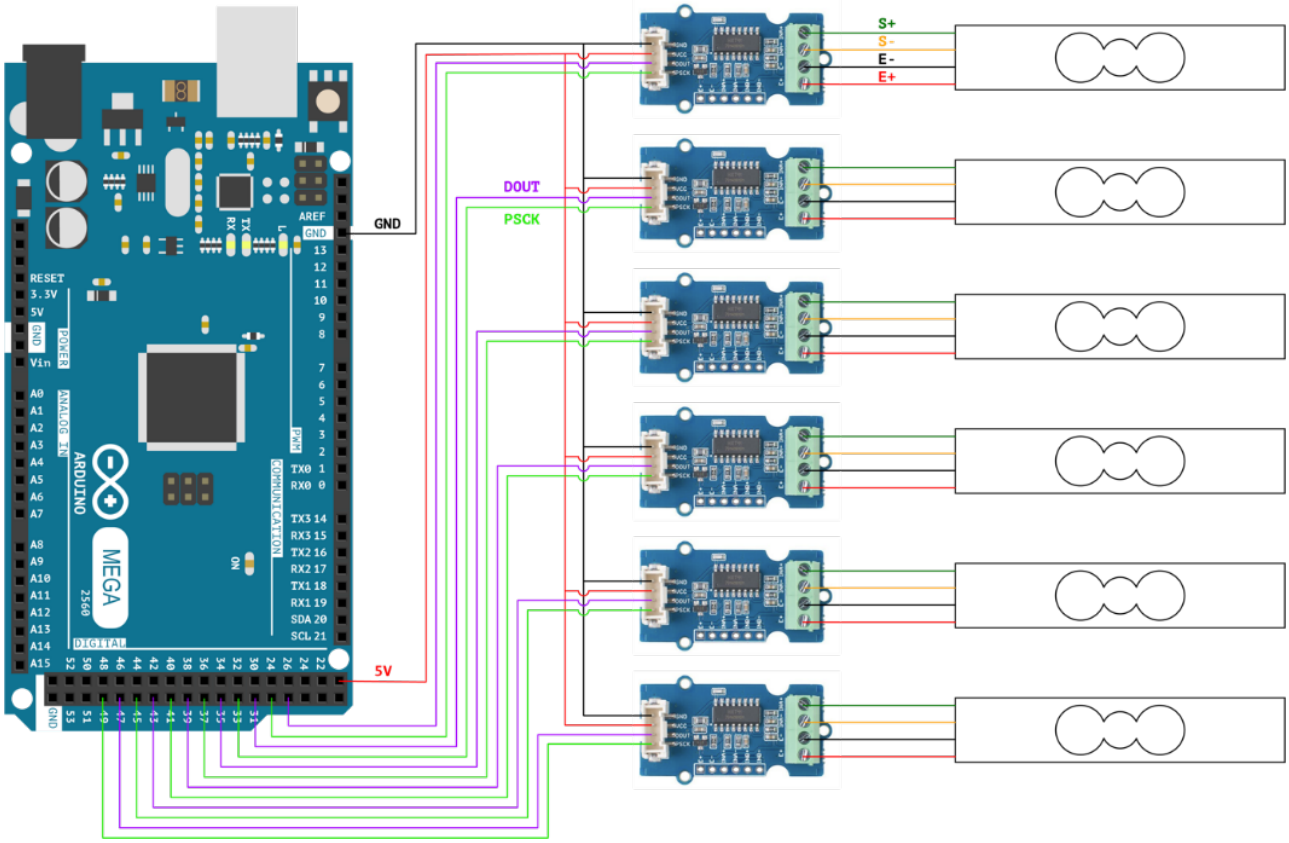


Figure 10 : Schéma de câblage permettant de récupérer les signaux des 6 capteurs d'efforts sur une carte Arduino Mega

III/ Etalonnage

Les modèles analytiques ou numériques utilisés pour le prédimensionnement sont très utiles pour dimensionner les composants, mais peuvent s'avérer trop imprécis lors de l'usage de la cellule de charge du fait, par exemple, des défauts de fabrication. Aussi, il est impératif de déterminer expérimentalement le modèle de comportement de chaque cellule 6 axes.

D'un point de vue pratique, la première étape a été de définir un repère sur le plateau supérieur de chacun des modèles dont l'origine est le centre du plateau (voir Figure 11). Ce repère permet d'exprimer les actions mécaniques exercées sur le plateau.

La seconde étape consiste à appliquer des chargements statiques, d'amplitude et direction connues, sur le plateau supérieur des cellules en des points particuliers. Pour maîtriser pleinement ce chargement, on utilise une masselotte de masse connue, suspendues au bout d'un câble accroché en un point particulier du plateau (voir Figure 12). En utilisant des poulies, on peut modifier l'orientation de l'effort appliqué. En faisant varier point d'application, masse, et position des poulies, on obtient toute une famille de chargements différents.

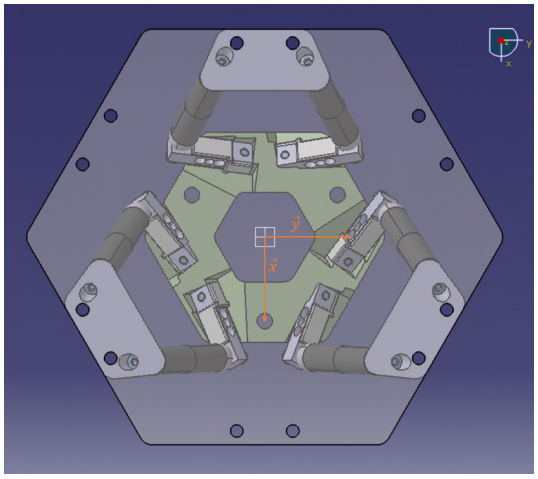


Figure 11 - Repère du modèle A

Figure 12 - Exemple d'application d'un glisseur sur le plateau du modèle B à l'aide d'une poulie et d'une corde rigide

IV/ Un exemple d'utilisation : mesure des actions mécaniques maximales transmissibles par le drone Snaptain A15

Dans l'industrie, de nombreux projets nécessitent de connaître les actions mécaniques transmissibles par un drone en plein vol. Ceci est alors possible en utilisant une cellule d'efforts 6 axes telles que celles présentées précédemment.

Le protocole consiste à encastrer un drone sur le plateau supérieur (voir Figure 14) de la cellule d'effort (modèle A en pratique) et de le piloter grâce à une télécommande de contrôle. Le but est d'abord de lui imposer une consigne de décollage puis des consignes de translation (horizontale ou verticale) ou de rotation (roulis, tangage ou lacet). Lorsque l'utilisateur envoie ces consignes de déplacement au drone encastré sur le plateau supérieur, celui-ci reçoit l'ordre de translater ou pivoter mais ses mouvements sont bloqués par l'encastrement. Le drone transmet alors au plateau supérieur les couples et les glisseurs qu'il est capable de produire selon ses axes de roulis, tangage ou lacet. D'un point de vue mathématique, cela revient à appliquer un torseur d'action mécanique au centre du plateau supérieur, ce qu'est capable de mesurer la cellule. Une simple analyse statique nous convainc que le plateau permet donc bien de mesurer les actions mécaniques que peut produire le drone lorsque celui-ci est en vol. Il est cependant nécessaire de faire le zéro des capteurs une fois l'encastrement du drone effectué pour s'affranchir du poids. Un schéma du montage est proposé en Figure 13.

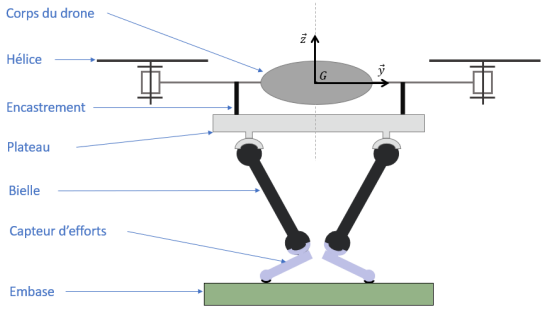


Figure 13 - Schéma du drone encastré sur le plateau supérieur de la cellule A

Pour réaliser un essai, nous allons utiliser un drone de la marque Snaptain et de type A15. Ce drone possède un faible coût et possède une masse faible de 167g, batterie incluse. Au vu de la masse de ce petit drone, l'encastrement a été réalisé avec du simple fil de fer de 1mm de diamètre. Cet encastrement rudimentaire a l'avantage d'être facilement reproductible, démontable, adaptable à d'autres drones de masse faible et possède un coût très faible. Le fil de fer est alors enroulé autour des bras du drone et introduit dans les trous présents sur le plateau supérieur pour relier le drone à ce dernier. Un socle imprimé 3D aurait pu également être conçu et fabriqué pour permettre un meilleur encastrement que celui assuré par le fil de fer. Cependant, il aurait fallu

concevoir un socle adaptable à d'autres types de drone, ce qui n'est pas chose aisée.

Figure 14 - Dispositif expérimental de mesure des performances d'un drone

Il est également important que le flux d'air généré par les hélices soit perturbé le moins possible par le plateau. Il faut donc choisir les dimensions du plateau de charge en fonction de l'envergure du drone à tester. L'étalonnage de la cellule d'effort a été réalisé en amont de l'essai. Le drone est piloté manuellement via sa télécommande au cours des différents essais réalisés.

Une fois que le drone a été installé sur le plateau, le protocole a été le suivant : relier la cellule d'effort à un ordinateur à l'aide du boîtier Arduino ; effectuer le zéro de la cellule d'efforts pour ne pas prendre en compte le poids du drone dans les mesures ; commander via la télécommande la mise en rotation des hélices du drone ; commander toujours avec la télécommande le décollage de ce dernier ; commander ensuite le déplacement du drone souhaité – translation à droite, à gauche, ascension verticale, etc. – ; relever et interpréter les mesures. A titre d'exemple, la poussée verticale maximale que peut produire le drone Snaptain A15 lors d'une consigne est affichée en Figure 156. Les 3 essais ont été réalisés dans des conditions similaires : pièce avec de l'air au repos et batterie chargée au maximum.

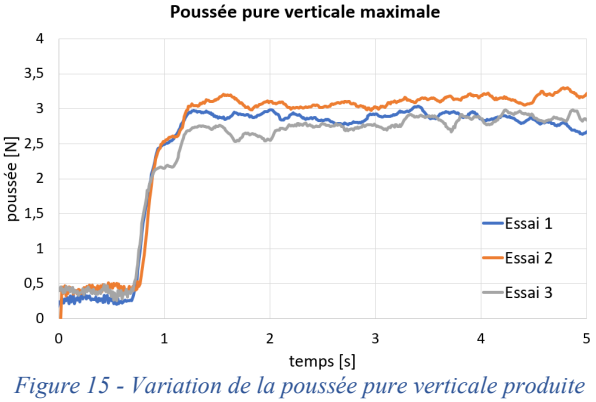


Figure 15 - Variation de la poussée pure verticale produite par le drone A15 lorsque l'utilisateur commande une ascension verticale au maximum des capacités du drone

Sur la Figure 15, est représentée la poussée verticale produite par le drone lorsque l'utilisateur commande une ascension verticale au maximum des capacités du drone (joystick de la télécommande en butée). La courbe ainsi obtenue se décompose en deux phases : une première où la poussée mesurée est de 0.4N en moyenne, qui correspond à la mise en rotation des hélices du drone au ralenti ne permettant pas de le faire décoller ; une seconde où la poussée augmente suite à une consigne d'ascension verticale à 0.59s. La poussée se stabilise à 3 N en moyenne ce qui est cohérent par rapport à la masse du drone. On note également une bonne répétabilité des mesures. Les écarts peuvent s'expliquer par plusieurs facteurs comme notamment la présence de frottements dans les liaisons entrainant une petite hystérésis.

V/ Conclusions et perspectives d'amélioration d'usage

Cet article a permis de développer la conception de deux cellules d'efforts 6 axes à moindre frais. Deux modèles avec des tailles et des solutions techniques différentes ont été proposés tout en maintenant un coût de fabrication inférieur à 150 euros. Dans chacun des cas, il a fallu trouver un compromis entre frottements et jeux des liaisons pour permettre des mesures correctes des efforts appliqués au plateau supérieur. Cette méthode de conception et de fabrication peut être adaptée en fonction du besoin en jouant sur la taille, la nature des liaisons mécaniques ou encore le système d'acquisition du système. Ce type de cellule 6 axes est également rapide à fabriquer (moins d'une semaine) et ne nécessite pas de moyens de production complexes. L'étalonnage est relativement simple et ne nécessite pas de manipulations complexes. A l'avenir, pour complètement valider cette conception, il faudra étudier la répétabilité des mesures ainsi qu'estimer la durée de vie des cellules ainsi construites. Une étude sur les performances dynamiques pourrait également être utile pour en connaître les limites.

MONTRE CONNECTÉE PROJET PÉDAGOGIQUE BLUETOOTH LOW ENERGY, STM32, APPLICATION SMARTPHONE XAMARIN

THOMAS MONGAILLARD
Élève en 3ème année de l'ENS Paris-Saclay

Résumé : *Ce sujet de mise en œuvre d'une connexion Bluetooth Low Energy entre une carte micro-contrôleur et un smartphone est à destination des enseignants en BUT GEII Option Électronique et Systèmes Embarqués dans le cadre d'un projet de Situation d'Apprentissage et d'Évaluation en 3ème année.*

Cette ressource présente les différentes étapes permettant la mise en œuvre d'une connexion Bluetooth Low Energy entre un micro-contrôleur et un smartphone. Elle s'appuie sur un projet de montre connectée permettant la transmission des données de pulsation cardiaque vers un smartphone. Une première partie expose les concepts principaux de la norme Bluetooth Low Energy. Ensuite, une deuxième partie détaille les étapes pour la programmation du microcontrôleur. Enfin, une troisième partie présente le développement d'une application Android avec l'outil Xamarin pour la réception des données sur smartphone.

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay

I/ Présentation du projet

Le but de ce projet est de réaliser une montre connectée communiquant avec un smartphone via une liaison Bluetooth Low Energy (BLE). La montre connectée se compose des éléments suivants :

- Un écran LCD ;
- Un capteur de pulsation cardiaque ;
- Un micro-contrôleur ;
- Une antenne ;
- Une batterie.

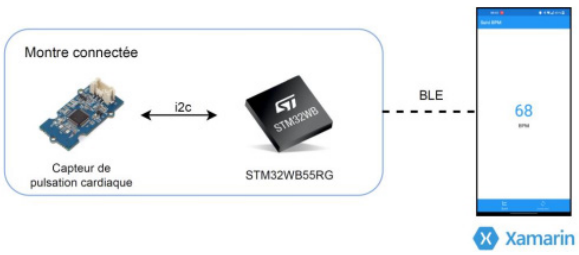


Figure 1 : Synoptique

Le système global doit pouvoir permettre d'effectuer les actions suivantes :

- Consulter l'heure sur la montre ;
- Consulter la pulsation cardiaque instantanée sur la montre ;
- Consulter l'évolution sur une heure de la pulsation cardiaque sur le téléphone ;
- Être alerté en cas de pulsation cardiaque anormale;
- Consulter l'évolution entière de la pulsation cardiaque en ligne.

Dans ce sujet, on s'intéresse particulièrement à la liaison entre le smartphone et la montre et à la visualisation de l'évolution de la pulsation cardiaque sur le smartphone de l'utilisateur (action en italique). Deux parties sont donc à traiter : le programme BLE sur le micro-contrôleur et l'application mobile

I.1/ Micro-contrôleur

On utilise la carte de développement STM32 NUCLEO-WB55RG[1] qui embarque le microcontrôleur STM32WB55RG[2] faisant partie de la gamme des micro-contrôleurs spécialisés pour les communications sans fils.

[1] https://www.st.com/en/evaluation-tools/nucleo-wb55rg.html

[2] https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-armcortex-mcus/stm32-wireless-mcus/stm32wb-series/stm32wbx5/stm32wb55rg.html

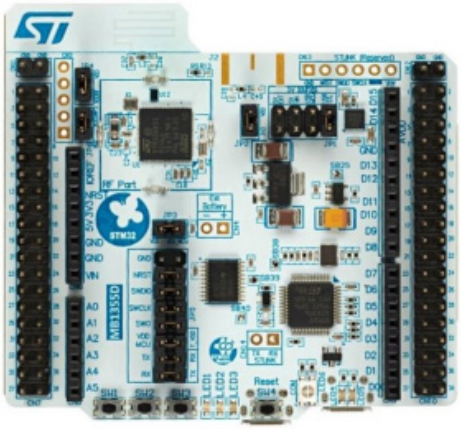


Figure 2 : Carte de développement Nucleo-WB55RG

Il intègre les normes Bluetooth 5.2 et ZigBee au sein d'un coprocesseur Arm® Cortex® M0+ couplé à un processeur Arm® Cortex® M4 sur lequel le programme hôte a lieu. Les fonctionnalités principales du STM32WB55RG sont :

- Émetteur/Récepteur Radio
- Modes d'ultra basse consommation
- Timers
- Mémoire Flash de 1 MB
- ADC 12 bits
- Protocoles USB, I2C, USART, SPI
- Algorithmes de chiffrement AES, RSA, Diffie-Helman.

La programmation de cette carte se fera par l'intermédiaire des outils de développement logiciels[3] développés par ST :

- L'environnement de développement : STM32 Cube IDE
- Le générateur de code : STM32 Cube MX
- Le gestionnaire de programmation et monitoring : STM32 Cube Programmer

I.2/ Capteur de pulsation cardiaque

Le capteur choisi ici est le Heart Rate Sensor de chez Seeed basé sur le capteur optique PAH8001EI-2G du fabricant PXI. Il possède une interface I2C pour une communication simple avec le microcontrôleur.

Figure 3 : Capteur de pulsation cardiaque

I.3/ Application Android et iOS

Pour pouvoir déployer une application mobile conjointement sur Android et iOS, on utilise l'outil Xamarin Forms développé par Microsoft et disponible sur l'environnement de développement Visual Studio.

Les projets sont écrits en langage C#, un langage dérivé du C++ et très proche du Java. Xamarin propose un kit de développement écrit en C# (regroupant des éléments graphiques, des structures de navigation, etc) et s'occupe de transformer le code associé en deux projets Android et iOS à la compilation. Il est possible de gérer indépendamment les deux projets au niveau des icônes et des couleurs par exemple.

L'application mobile doit permettre les actions suivantes :

- Établir la connexion entre le téléphone et la montre;
- Recevoir les données de pulsation cardiaque ;
- Afficher en temps réel l'évolution de la pulsation cardiaque de l'utilisateur ;
- Envoyer les données de pulsation cardiaque sur un serveur de base de données.

La mise en œuvre de la dernière action n'est pas détaillée dans cette ressource.

II/ Bluetooth Low Energy

Certaines des figures de cette sous-section proviennent du document STM32WB BLE programming guidelines[4], ainsi que du site de Bluetooth[5].

II.1/ Modélisation et séparation des fonctions

L'architecture BLE et son implémentation se décompose principalement en deux parties

[3] https://www.st.com/content/st_com/en/products/development-tools/software-developmenttools/stm32-software-development-tools.html

[4] https://www.st.com/resource/en/programming_manual/pm0271-stm32wb-ble-stack-programmingguidelines-stmicroelectronics.pdf

[5] https://www.bluetooth.com/

fonctionnelles : un contrôleur et un hôte. On représente la pile sous une forme de modèle en couche type OSI en figure 4.

Le contrôleur est composé de la liaison physique ainsi que du Link Layer. L'hôte gère la mise en forme des données et la gestion des connexions. On peut le décomposer par ses différentes fonctions : l'adaptation protocole (L2CAP), le security manager (SM), l'attribute protocol (ATT), le Generic Attribute profile (GATT) et le Generic Access Profile (GAP). Le contrôleur et l'hôte communiquent entre eux via l'interface HCI (Host Controller Interface). Ces différentes couches sont détaillées dans la suite.

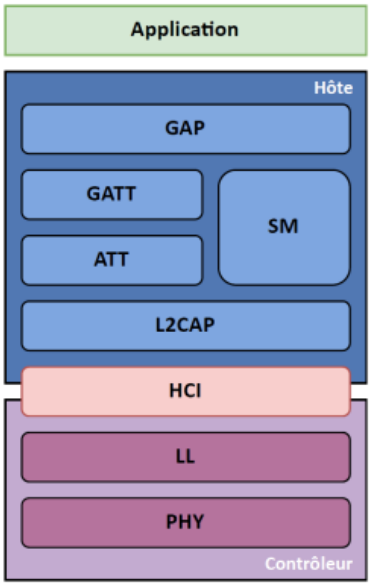


Figure 4 : Pile BLE

II.2/ Modes de communication

La norme Bluetooth Low Energy supporte plusieurs topologies de communication possibles présentées en figure 5.

Figure 5 : Topologies de communication

Le mode point à point est le mode le plus utilisé dans l'utilisation quotidienne. Ce mode est utilisé pour connecter deux appareils (une souris sans fil à un ordinateur par exemple). Le mode broadcast est utilisé dans des architectures plus grandes où certains appareils diffusent des données à d'autres appareils qui écoutent aux alentours. Il n'y a donc pas de connexion nécessaire entre les appareils et ce mode trouve ses applications dans les systèmes de mesures (un capteur diffuse sa mesure) ou dans la géolocalisation en intérieur (des balises indiquent leurs positions dans un bâtiment). Enfin, le réseau mesh est moins utilisé et plus récent que les autres. Dans cette topologie, chaque appareil est à la fois client et serveur de telle sorte que tous les appareils peuvent communiquer entre eux.

II.3/ Connexion et appairage dans le mode point à point

La connexion entre deux appareils BLE dans le mode point à point se réalise en plusieurs temps. On résume les interactions nécessaires à la connexion dans le diagramme de séquence en figure 6. Ce mode met en jeu deux rôles :

- L'appareil central pouvant scanner autour de lui pour détecter des dispositifs BLE et initier une connexion ;
- L'appareil peripherial pouvant indiquer sa présence, accepter des connexions, mais pas les initier.

Ces rôles sont configurés dans les appareils à travers le Generic Access Profile (GAP). Ce profil permet aussi la configuration des options de connexion.

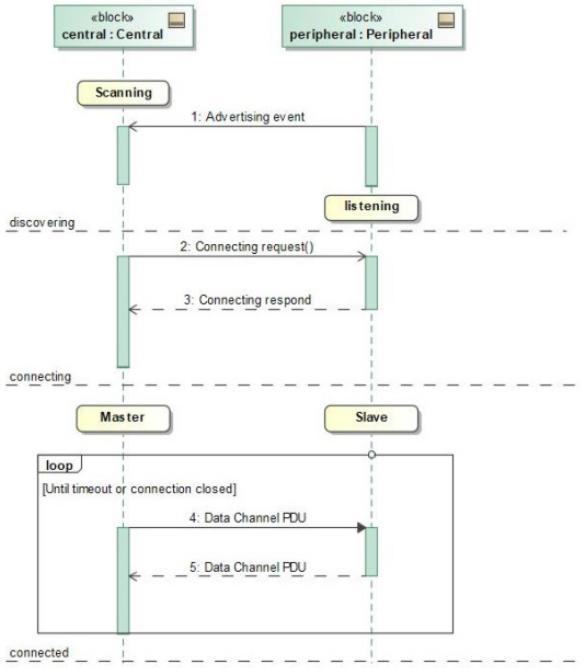


Figure 6 : Diagramme de séquence de connexion

Dans un premier temps, les deux appareils ne se connaissent pas. Un appareil peripherial qui cherche à communiquer va envoyer périodiquement des trames d'annonce (advertisment) qui contiennent des informations sur l'émetteur, sa fonction et sur la connexion. Après chaque émission, le périphérique va attendre une réponse avant d'émettre à nouveau. Après détection du message d'annonce et s'il y a volonté de se connecter, l'appareil central va envoyer une demande de connexion qui va être acquittée par le peripherial. On peut alors communiquer les informations ou demandes d'informations souhaitées, et ce, jusqu'à ce que la connexion soit terminée. On parle alors de maître et d'esclave comme dans une liaison I2C.

II.4/ Statut d'un périphérique (Link Layer)

On peut définir le statut d'un périphérique par différents états qui relatent de la connexion. On présente le graphe d'état d'un appareil en figure 7.

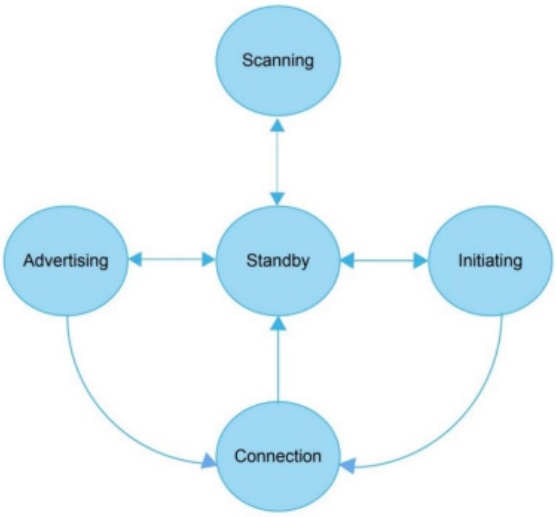


Figure 7 : Graphe d'état d'un périphérique

On explicite les différents états :

• Standby : traduit l'absence de flux de données entrant ou sortant ;

• Advertising : traduit la diffusion d'un message d'annonce ;

• Initiating : traduit l'initiation d'une connexion ;

• Connection : traduit l'échange de données en tant que maître/esclave ;

• Scanning : traduit l'attente et l'écoute de messages d'annonce.

II.5/ Paquets BLE

Les trames envoyées et reçues ont toujours la même structure, à savoir celle représentée sur la figure 8.

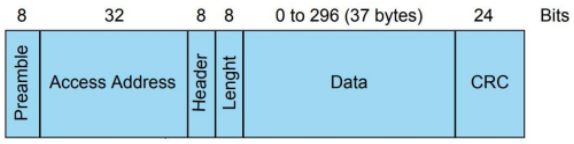


Figure 8 : Structure trame

On détaille les différents champs :

• Le champ *Preamble* est une séquence permettant la synchronisation radiofréquence ;

• Le champ *Access Address* est une adresse propre au contenu de la trame émise/demandée ;

• Le champ *Header* permet de définir le type de la trame (publicité, connexion, réponse à un scan, etc.) ;

• Le champ *Length* permet d'indiquer la quantité d'octets à lire ;

• Le champ *Data* contient les données transmises sous la forme d'un tableau d'octets ;

• Le champ *CRC* permet une protection des données contre les erreurs de bits lors de la transmission.

II.6/ Attribute Protocole

Lorsque l'on transmet des données en BLE, on utilise l'Attribute protocole (ATT). Ce protocole définit une structure pour la communication des données. Ces dernières prennent la forme d'attributs et sont définis par:

• Sa référence ;

• Son type défini par un Universally Unique IDentifier (UUID) ;

• Sa valeur ;

• Les permissions d'écriture et de lecture associées.

On appelle :

• Serveur l'appareil possédant les attributs à communiquer ;

• Client l'appareil demandant l'accès à des attributs.

Par exemple, si on donne l'accès à une température en lecture uniquement et sans sécurité, on aurait :

Attribute handle	Attribute type	Attribute value	Attribute permissions
0x0008	"Temperature UUID"	"Temperature Value"	"Read only, no authorization, no authentication"

Figure 9 : Exemple d'attribut

Protocol data unit (PDU message)	Sent by	Description
Request	Client	Client asks server (it always causes a response)
Response	Server	Server sends response to a request from a client
Command	Client	Client commands something to server (no response)
Notification	Server	Server notifies client of new value (no confirmation)
Indication	Server	Server indicates to client new value (it always causes a confirmation)
Confirmation	Client	Confirmation to an indication

Figure 10 : Description des PDU

Il est possible d'interagir de plusieurs manières avec un serveur. On définit alors plusieurs protocoles de communication sur les attributs nommés Protocol Data Unit (PDU) en figure 10. On peut voir cette liaison client/serveur de la même manière qu'une base de données avec un client qui vient interagir avec des accès plus ou moins restreints à des tables de valeurs. Dans le message d'annonce transmis lors de la phase advertising, il est possible d'introduire un attribut représentant la fonction de l'appareil, permettant alors certains systèmes d'exploitation d'indiquer à l'utilisateur quel type d'objet a été scanné. Par exemple, ils affichent un logo de casque lorsqu'un casque Bluetooth est scanné.

II.7/ Generic Attribut Profile (GATT)

Afin d'utiliser plus simplement les attributs, on passe par une structure utilisant le protocole ATT.

Cela permet de simplifier les échanges d'attributs. En effet, ATT ne concerne qu'une unité de donnée alors que le GATT permet l'encapsulation de plusieurs attributs. Les GATT suivent également le format de la trame.

Il existe 3 types d'attributs :

• *Characteristic* : représente une donnée,

• *Descriptor* : décrit plus en détail une *characteristic*,

• *Service* : regroupe plusieurs *characteristics* pour un service particulier.

II.8/ Bluetooth Special Interest Group (SIG)

Il existe des spécifications/normes écrites par le Bluetooth SIG à suivre pour différentes applications qui sont des profils standards de Bluetooth. Les profils sont détaillés dans ces normes et comportent entre autres les rôles du GAP, les services présents, les paramètres de connexion.

Les services eux-mêmes sont détaillés dans un document à part. Tous ces documents sont disponibles sur ce lien[6]. Il existe un profil pour les dispositifs de mesure de pulsation cardiaque appelé *Heart Rate Profile* qui correspond parfaitement à notre application.

III/ Projet STM32

III.1/ Création d'un projet d'exemple pour WB55RG

L'outil CubeMX permet de générer un projet à partir des exemples fournis par ST. Pour notre projet, on va utiliser l'exemple BLE_HeartRate qui répond à la spécification du Bluetooth SIG mentionné précédemment. CubeMX peut générer l'exemple pour différents IDE, on utilisera ici CubeIDE.

Les captures d'écran suivantes montrent les étapes à suivre :

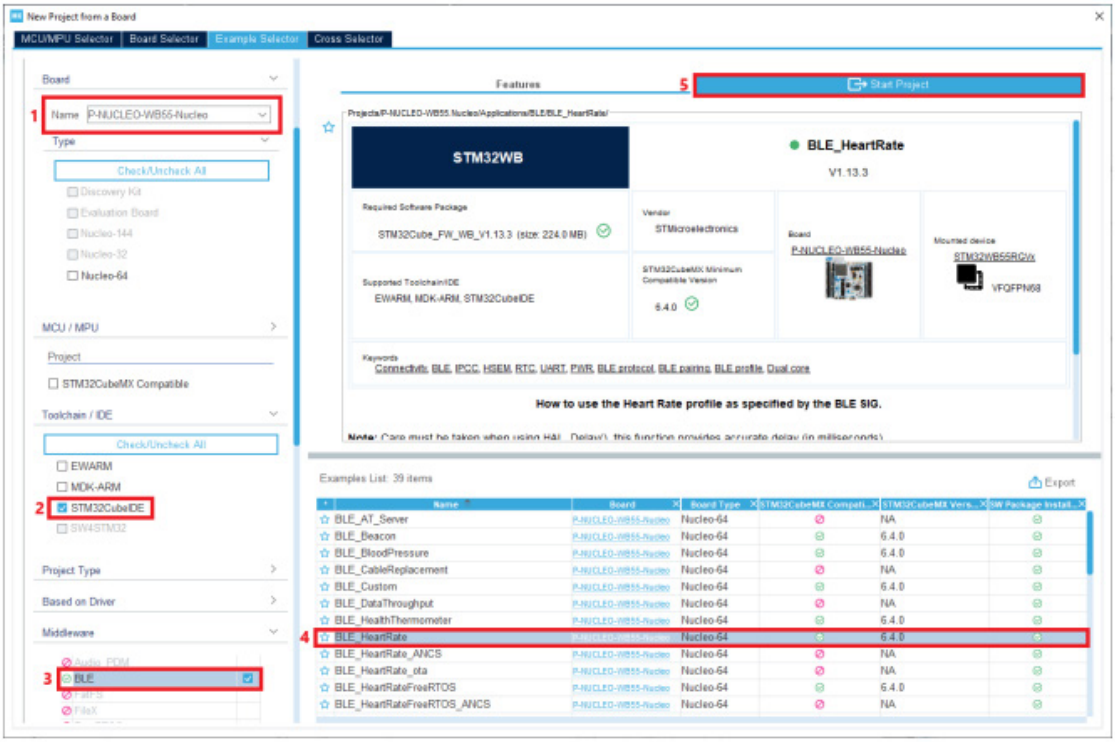


Figure 11 : Sélection de l'exemple

Le code a été généré, on lance alors CubeIDE. Pour importer le projet créé, aller dans **File → Import → General → Existing Projects into Workspace** puis sélectionner le dossier dans le lequel 9 a été créé le projet. Il est alors possible de sélectionner le projet BLE_HeartRate puis cliquer sur **Finish**.

Le fichier principal main.c se trouve dans **Application → User → Core**. L'en-tête de ce fichier précise qu'il est nécessaire que le Wireless Coprocessor soit flashé avec un certain fichier binaire grâce à CubeProgrammer.

III.2/ Flash du Wireless Coprocessor

Les fichiers binaires nécessaires pour mettre à jour le coprocesseur se trouvent sur cette page[7]. Télécharger le paquet Patch-CubeWB puis le dézipper. Lancer à présent CubeProgrammer en ayant branché la carte de développement et suivre les étapes suivantes :

1. Déterminer quelle version du Firmware Upgrade Service (FUS) est installée (figure 14)
2. Mettre à jour le FUS :
   a. Cliquer sur Browse
   b. Depuis le dossier dézippé, aller dans Projects/STM32WB_Copro_Wireless_Binairies/STM32WB5x/ et ouvrir
      - stm32wb5x_FUS_fw_for_fus_0_5_3 si la version du FUS est 0.5.3
      - stm32wb5x_FUS_fw sinon

[6] *https://www.bluetooth.com/specifications/specs/*

[7] *https://www.st.com/cas/login*

c. Vérifier que le champ Start address vaut 0x080EC000

d. Cliquer sur Firmware Upgrade

En cas d'erreur, vérifier que le fichier ouvert est bien conforme à la version du FUS et que l'adresse de départ est correcte. En cas de version micro-contrôleur différente, consulter le fichier Release_Notes.html présent dans le même dossier que précédemment et déterminer la bonne adresse de départ.

3. Mettre à jour le stack BLE :

a. Cliquer sur Browse

b. Ouvrir le fichier stm32wb5x_BLE_Stack_full_fw dans le même dossier que précédemment

c. Vérifier que le champ Start address vaut 0x080C7000

d. Cliquer sur Firmware Upgrade

Le coprocesseur a été mis à jour, on va pouvoir exécuter l'exemple sur la carte.

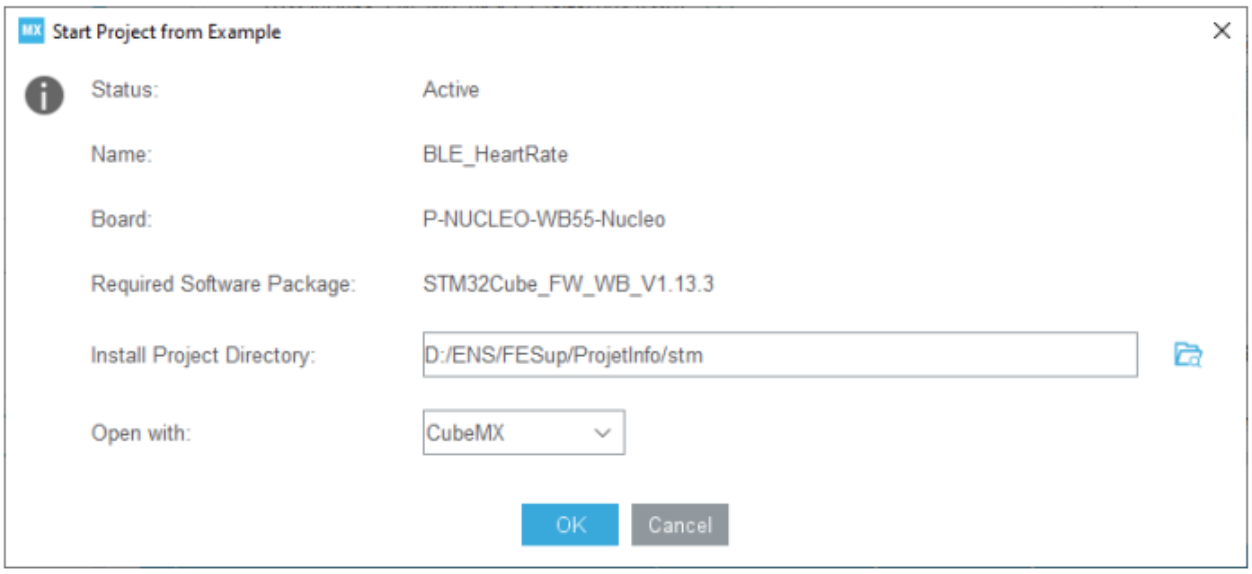


Figure 12 : Création du projet

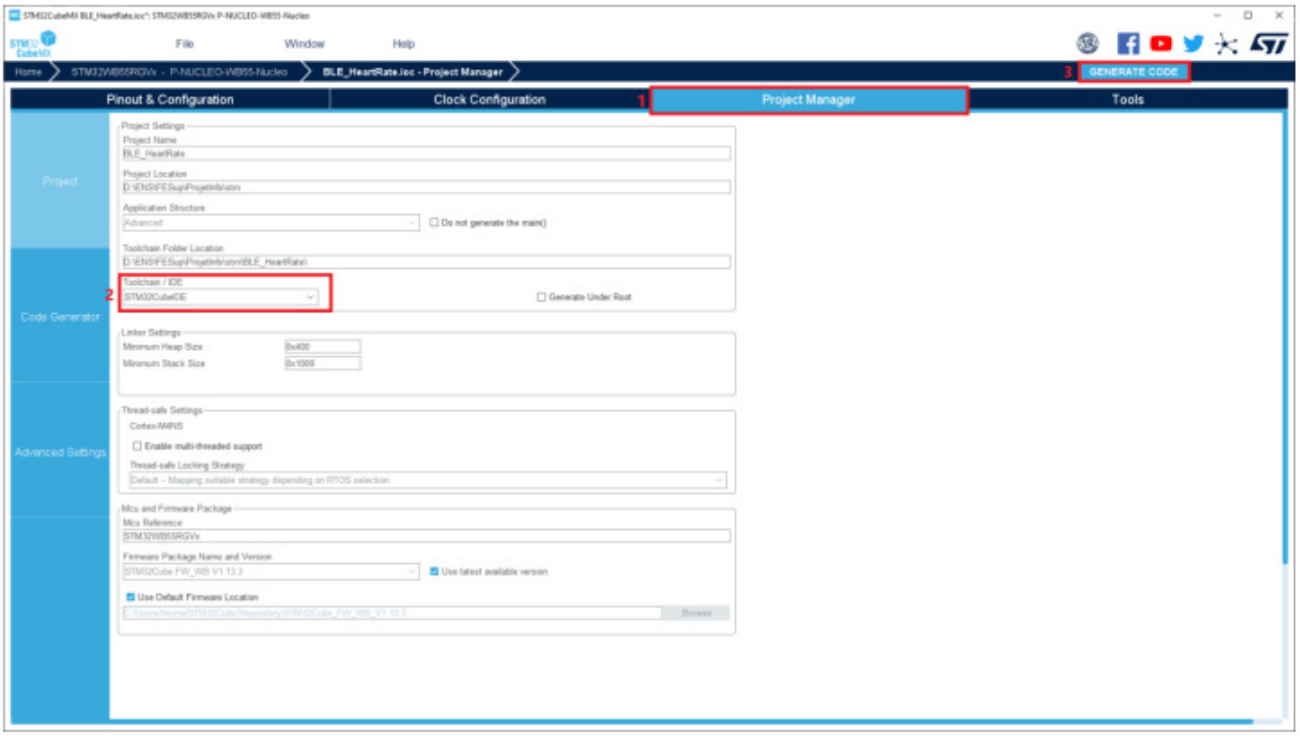


Figure 13 : Génération du code

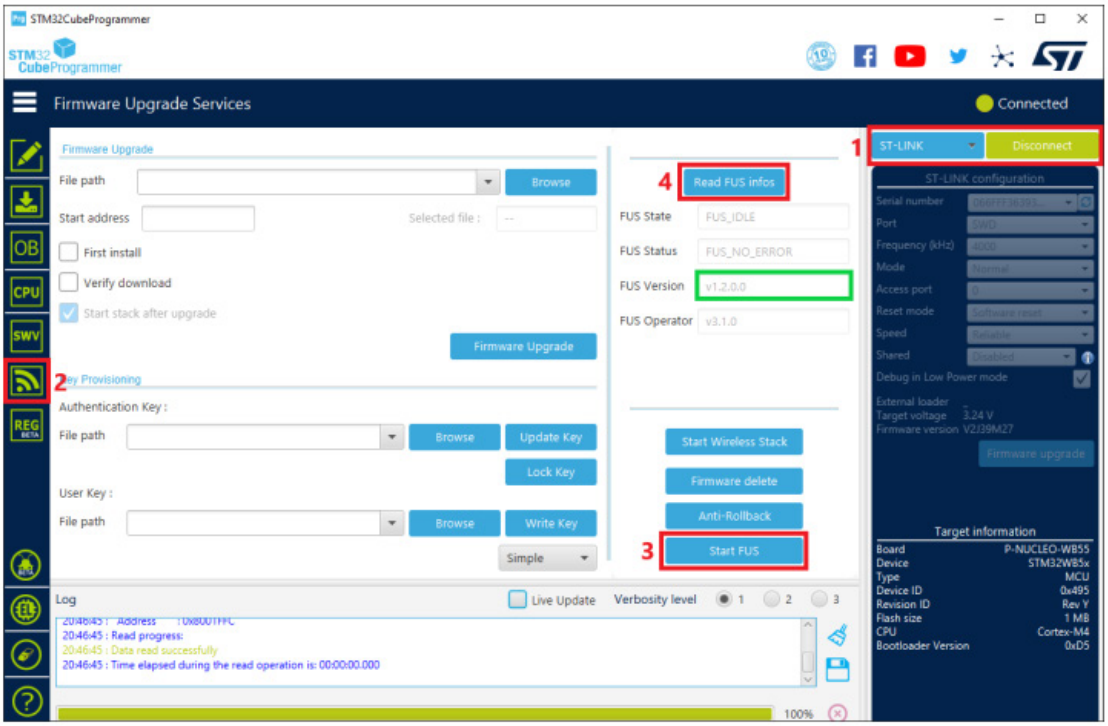


Figure 14 : Version du FUS

III.3/ Analyse de l'exemple

III.3.1/ Exécution du code

Par défaut, le débogage de la partie BLE n'est pas activé. Pour l'activer, double-cliquer sur le fichier BLE_HeartRate.ioc dans l'explorateur de projet à gauche puis suivre les étapes de la capture de la figure 15. Une fois terminé, enregistrer le fichier .ioc (Ctrl+S) pour regénérer le code.

On peut maintenant lancer le build (Ctrl+B) puis le débogage (F11). Le code étant téléversé sur la carte, on peut vérifier son bon fonctionnement avec l'application mobile ST BLE Profile (figure 17).

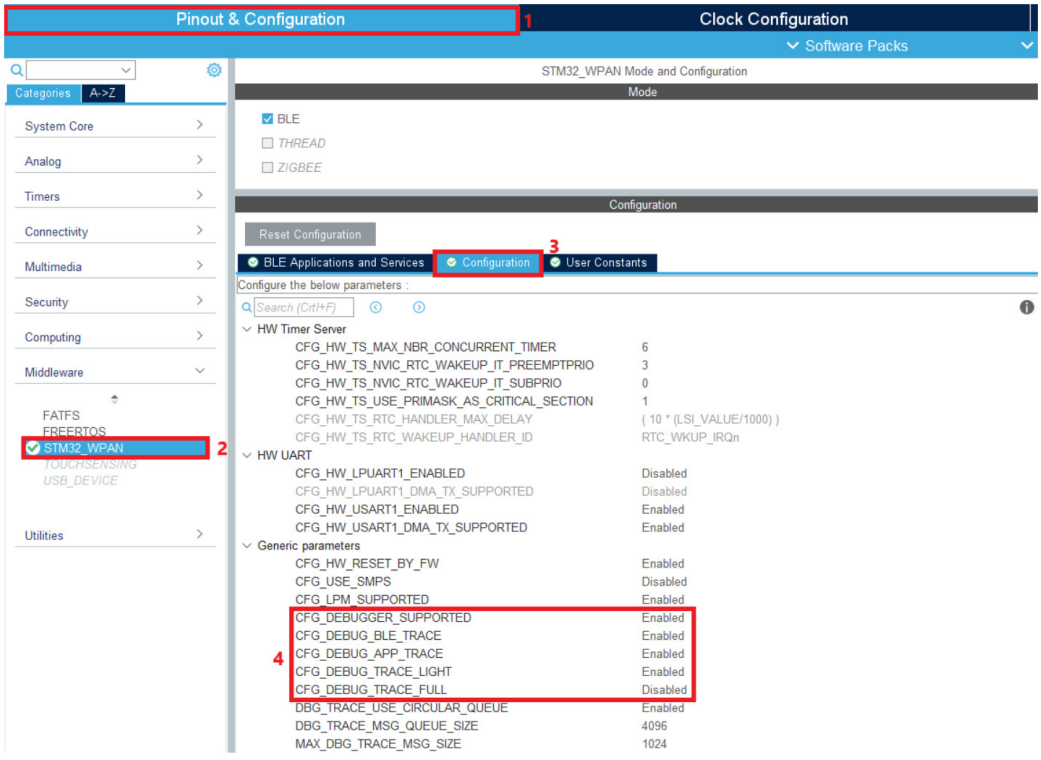


Figure 15 : Activation du débogage

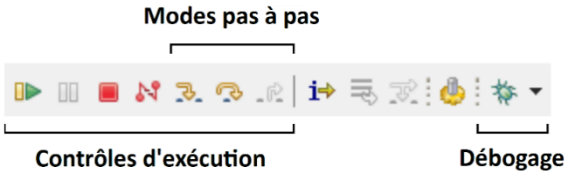


Figure 16 : Barre d'outils CubeIDE

On peut voir qu'une fois la connexion effectuée, on reçoit à intervalle régulier court (< 1s) une valeur de pulsation cardiaque en battement par minute. Il n'y a pas besoin de demander depuis l'application une nouvelle mesure puisque c'est le micro-contrôleur qui notifie le téléphone à chaque nouvelle mesure. On remarque sur la figure 17 que l'on obtient une courbe de pulsation cardiaque périodique alors qu'aucun capteur n'a encore été branché. Il s'agit donc de mesures arbitraires générées par l'exemple pour montrer le fonctionnement. Il s'agira par la suite de modifier le code associé pour le remplacer par la mesure réelle provenant du capteur de pulsation cardiaque.

III.3.2/ Configuration BLE

Le fichier qui permet de configurer la communication est Application/User/STM32_WPAN/app_ble.c. La fonction principale est APP_BLE_Init dans laquelle on s'intéresse aux fonctions suivantes :

• Ble_Hci_Gap_Gatt_Init : initialise les couches HCI, GAP et GATT ;

• SVCCTL_Init : initialise les profils Bluetooth ;

• HRSAPP_Init : initialise les actions associées au profil Heart Rate.

Pour rentrer dans le code d'une fonction, utiliser le raccourci F3.

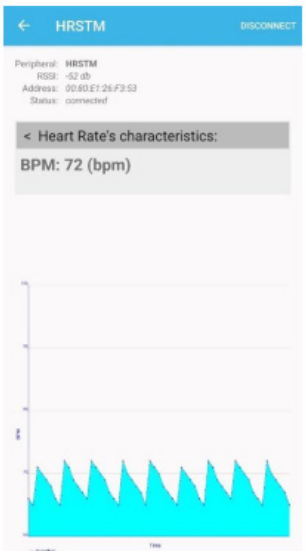


Figure 17 : Application ST BLE Profile

Initialisation GAP Dans Ble_Hci_Gap_Gatt_Init, on définit le rôle du GAP du côté du capteur, à savoir le rôle Peripherial puisqu'il est l'appareil avec le moins de ressources (calcul et autonomie). On définit en plus le nom du capteur HRSTM qui est lu lors du scan par le téléphone ainsi que l'apparence (figure 18).

```
  /**
   * Initialize GAP interface
   */
  role = 0;

#if (BLE_CFG_PERIPHERAL == 1)
  role |= GAP_PERIPHERAL_ROLE;
#endif

#if (BLE_CFG_CENTRAL == 1)
  role |= GAP_CENTRAL_ROLE;
#endif

  if (role > 0)
  {
    const char *name = "HRSTM";
    aci_gap_init(role,
#if ((CFG_BLE_ADDRESS_TYPE == RESOLVABLE_PRIVATE_ADDR) || (CFG_BLE_ADDRESS_TYPE == NON_RESOLVABLE_PRIVATE_ADDR))
                 2,
#else
                 0,
#endif
                 APPBLE_GAP_DEVICE_NAME_LENGTH,
                 &gap_service_handle,
                 &gap_dev_name_char_handle,
                 &gap_appearance_char_handle);

    if (aci_gatt_update_char_value(gap_service_handle, gap_dev_name_char_handle, 0, strlen(name), (uint8_t *) name))
    {
      BLE_DBG_SVCCTL_MSG("Device Name aci_gatt_update_char_value failed.\n");
    }
  }

  if(aci_gatt_update_char_value(gap_service_handle,
                                gap_appearance_char_handle,
                                0,
                                2,
                                (uint8_t *)&appearance))
  {
    BLE_DBG_SVCCTL_MSG("Appearance aci_gatt_update_char_value failed.\n");
  }
```

Figure 18 : Initialisation du GAP

GATT Services Les deux services Bluetooth associés au profil Heart Rate sont le Heart Rate Service et le Device Information Service. Celui qui nous intéresse ici est le Heart Rate Service dont la déclaration des attributs est faite dans le fichier Middlewares/STM32_WPAN/hrs.c. La fonction HRS_Init permet d'ajouter le service avec les caractéristiques associés et définies dans la norme du Bluetooth SIG. La seule caractéristique qui va nous intéresser ici est le Heart Rate Measurement, les autres étant inutiles pour notre application. Pour les désactiver, lancer le fichier .ioc et suivre les étapes de la capture de la figure 19. On remarque que la caractéristique Heart Rate Measurement a la propriété NOTIFY.

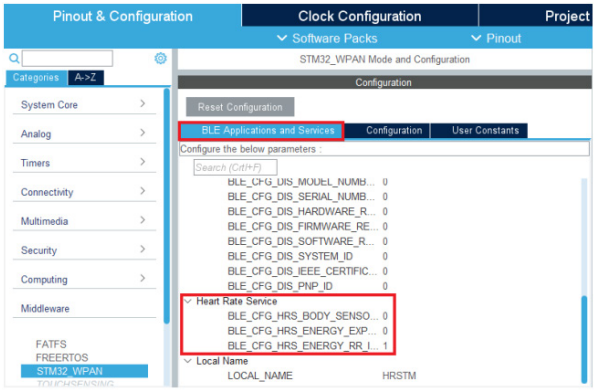


Figure 19 : Désactivation des caractéristiques inutiles

Notification La gestion de la notification est faite dans le fichier qui contient la fonction HRSAPP_Init. On remarque qu'un timer est créé à l'initialisation permettant de lancer la fonction HrMeas. Ce timer est lancé lorsque l'utilisateur demande la notification depuis le téléphone après s'être connecté, comme on peut le voir dans la fonction HRS_Notification. Lors du lancement du timer, on indique la période du timer HRSAPP_MEASUREMENT_INTERVAL.

La fonction HrMeas permet de lancer la tâche qui a été enregistrée à la première ligne de la fonction d'initialisation, permettant elle-même d'exécuter la fonction HRSAPP_Measurement. C'est dans cette fonction que l'on effectue la mesure arbitraire avant de mettre à jour la caractéristique pour permettre la notification vers le téléphone. Les valeurs de pulsation cardiaque sont générées par la ligne :

measurement = ((HRSAPP_Read_RTC_SSR_SS()) & 0x07) + 65; La fréquence des notifications n'est pas explicite dans le code puisqu'elle est définie à partir d'une horloge provenant d'une division de celle de l'oscillateur. On peut tout de même la modifier en augmentant ou diminuant le facteur dans la définition de HRSAPP_MEASUREMENT_INTERVAL.

III.4/ Ajout du capteur de pulsation cardiaque

On souhaite maintenant ajouter le capteur pour envoyer des vraies mesures au téléphone.

III.4.1/ Configuration I2C

Lancer le fichier .ioc et suivre les étapes de la figure 20.

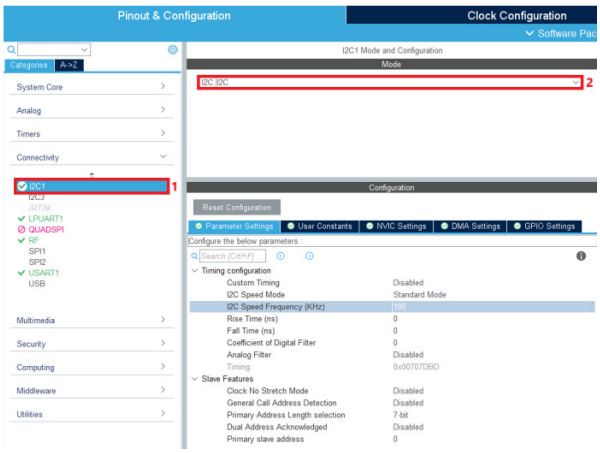


Figure 20 : Activation de la communication I2C

Dans l'onglet GPIO Settings, les pins associés aux signaux SCL et SDA sont PB8 et PB9. La figure 21 indique que les pins correspondantes sont nommées D15 et D14 sur la carte. Enregistrer le fichier pour regénérer le code. Dans le fichier main.c, une variable hi2c1 a été créée pour permettre la gestion de la communication I2C grâce à la bibliothèque HAL_I2C. La fonction d'initialisation a été générée automatiquement, il suffit maintenant d'écrire la fonction permettant de lire la valeur du capteur.

On peut alors brancher le capteur à la carte avec d'un côté l'alimentation en 5V et de l'autre les 2 signaux I2C. On s'assure que le capteur est bien alimenté en observant la DEL du capteur optique s'allumer.

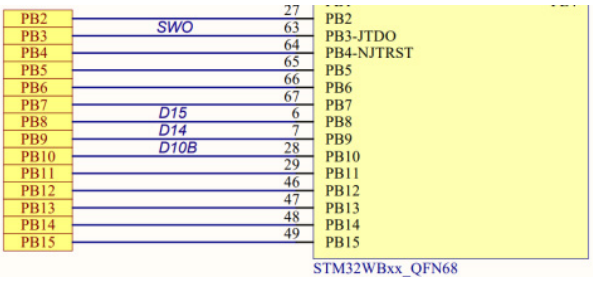


Figure 21 : Noms des pins Nucleo-WB55RG

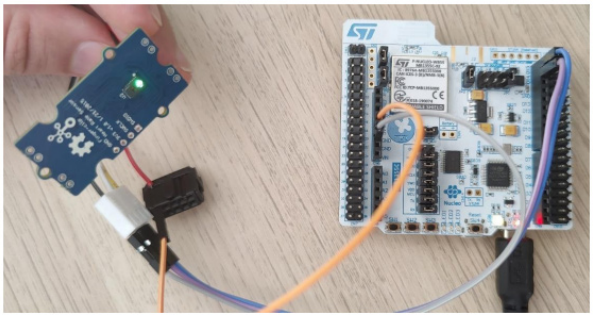


Figure 22 : Montage

III.4.2/ Récupérer la donnée capteur

La variable hi2c1 étant privée, on doit écrire cette fonction à l'intérieur du fichier main.c. En bas de celui-ci, une section USER CODE BEGIN 4 permet d'écrire des fonctions utilisateur sans qu'elles ne soient effacées lors d'une regénération de code. Le capteur renvoyant un unique octet représentant la valeur de la pulsation cardiaque, le prototype de la fonction sera uint8_t GetHeartRateValue(void). Au préalable, on définit deux variables dans la section USER CODE BEGIN PV :

- uint8_t HeartRateValue : permet de stocker la valeur renvoyée par le capteur
- uint16_t SensorAddress = 0xA0 : adresse I2C du capteur

On écrit ensuite la fonction :

```
// Lecture en I2C uint8_t GetHeartRateValue()
{
HAL_I2C_Master_Receive(&hi2c1,SensorAddress,&
HeartRateValue,1,500); return HeartRateValue;
}
```

La fonction HAL_I2C_Master_Receive prend en paramètre :

- Le handle de la liaison I2C ;
- L'adresse de l'appareil subordonné ;
- Le pointeur de la variable de stockage ;
- La taille du mot à lire en octet ;
- Le timeout. Ne pas oublier de rajouter le prototype de la fonction dans le fichier main.h à la section USER CODE BEGIN EFP.

On va peut remplacer la ligne de génération de valeur dans la fonction HRSAPP_Measurement du fichier hrs_app.c par :

measurement = GetHeartRateValue(); en incluant main.h en en-tête de fichier.

IV/ Application mobile

IV.1/ Création d'un projet Xamarin Forms

Capsule vidéo :

- • 01 : Création d'un projet Xamarin Forms pour le développement d'applications Android et iOS sous le format d'application à onglets.

IV.2/ Interface utilisateur Capsules vidéo :

- 02 : Utilisation du rechargement à chaud, permettant de modifier des caractéristiques graphiques dans le code et de visualiser en même temps
- 03 : Modification des icônes des onglets
- 04 : Nettoyage des fichiers non utiles provenant de l'exemple de base
- 05 : Ajout d'un bouton de scan sur la page de connexion dans le fichier .xaml avec sa fonction de callback
- 14 : Mise en page de la vue de suivi pour afficher la pulsation cardiaque

IV.3/ Connexion avec le micro-contrôleur

Capsules vidéo :

- 06 : Pour utiliser les fonctionnalités Bluetooth LE, on utilise le package Plugin.BLE[8]. On va pouvoir utiliser facilement des fonctions pour les différentes procédures LE
- 07 : Ajout des autorisations android dans le fichier manifest.xml
- 08 : Fonction de callback pour le scan
- 09 : Fonction qui se déclenche lors d'une détection d'appareil : on ne garde que les appareils LE possédant un nom et le service Heart Rate
- 10 : Correction du code pour l'affichage
- 11 : Requête de connexion après avoir trouvé un appareil compatible
- 12 : Découverte des services et de la caractéristique de mesure de la pulsation cardiaque
- 13 : Affichage des informations de connexion

IV.4/ Réception et affichage des données Capsules vidéo :

- 15 : Liaison des données avec l'activation des notifications et la fonction qui s'exécute lors d'une réception. Une modification graphique doit passer nécessairement par le thread principal
- 16 : Correction du code, il faut initialiser le tableau
- 17 : Ajout des autorisations qui doivent être acceptées par l'utilisateur.

IV.5/ Export et test

Capsules vidéo :

- 18 : Export de l'application android
- 19 : Test sur téléphone

Le projet sous iOS doit subir les mêmes étapes, il faut se référer à la documentation de Plugin.BLE pour ce qui concerne les autorisations.

V/ Références :

[1] Site Web Bluetooth, https://www.bluetooth.com/

[2] PM0271, STM32WB BLE stack programming guidelines, Programming manual, ST, https://www.st.com/resource/en/programming_manual/pm0271-stm32wb-ble-stackprogramming-guidelines-stmicroelectronics.pdf

[3] Documentation Xamarin, https://docs.microsoft.com/fr-fr/xamarin/get-started/what-isxamarin

[4]: Github Plugin.BLE, https://github.com/xabre/xamarin-bluetooth-le

[8] https://github.com/xabre/xamarin-bluetooth-le

MUTUALISATION DES RESSOURCES IOT PAR CONTENEURISATION DE PASSERELLE

SYLVAIN LEFEBVRE, MAHER JRIDI
L@bISEN, Vision-AD, ISEN Nantes, 33 Q, avenue du champ de manœuvre, 44000 Carquefou
sylvain.lefebvre@isen-ouest.yncrea.fr, maher.jridi@isen-ouest.yncrea.fr

Résumé : *La plateforme FIT IoT Lab est une plateforme académique ouverte permettant d'accéder à un ensemble de nœuds programmables à distance, pour l'enseignement ou les expérimentations en IoT. La publication de la conception et de la plupart des composants logiciels de cette plateforme sous licence libre permet à tous de concevoir et déployer une infrastructure compatible avec cet outil. Ce travail présente un nouveau type de nœud passerelle compatible avec l'infrastructure FIT IoT reposant sur des conteneurs logiciels. L'utilisation de conteneurs permet de réduire le nombre de composants matériels nécessaires à un déploiement.*

I/ Introduction

La pandémie de ces dernières années a poussé plusieurs établissements à se doter de moyens matériels et logiciels pour permettre un accès à distance à certains de leurs équipements éducatifs ou de recherche.

Certaines plateformes scientifiques ouvertes comme la plateforme FIT IoT LAB [1] permettent l'enseignement à distance de la programmation de microcontrôleurs pour des applications orientées IoT (internet des objets). La mise à disposition sous licence libre de la plupart des composants logiciels de cette plateforme permet à tous de déployer une infrastructure similaire. Les partenaires du FIT IoT LAB encouragent, par la même occasion, que les nouveaux sites soient équipés de nouveaux nœuds IoT hétérogènes pour enrichir la gamme de nœuds et les cas d'usage qui peuvent être associés. Cependant, l'achat de nombreux composants est nécessaire pour la mise en place de cette architecture et l'accès à certains d'entre eux est limité en raison de la crise qui frappe l'industrie des semi-conducteurs.

En effet, la crise sanitaire a aussi eu un impact sur les chaines d'approvisionnement en semi-conducteurs ce qui a réduit les quantités de matériels disponibles à l'achat pour construire de nouvelles plateformes.

Nous proposons dans ce travail une nouvelle architecture de nœud compatible avec la plateforme FIT IoT LAB. Cette approche permet de mutualiser le matériel nécessaire à la construction d'une passerelle en la mettant au service de plusieurs nœuds IoT. La quantité de matériel nécessaire au déploiement est réduite grâce à l'utilisation de la virtualisation au sein de la passerelle ce qui permet d'avoir accès à plusieurs nœuds IoT. Ainsi, nous ne proposons pas un nouveau matériel pour s'intégrer à la plateforme FIT IoT LAB, mais nous introduisons un nouvel agencement de nœuds en lien avec la passerelle pour permettre la mutualisation des ressources. En plus de l'avantage économique, ce nouvel outil peut enrichir les cas d'usage de plateforme FIT IoT LAB. En effet, en déployant plusieurs nœuds IoT séparés les uns par rapport aux autres de quelques centimètres, nous pourrons envisager résoudre des problématiques de communications numériques comme l'estimation du canal de transmission, la haute précision pour la localisation indoor ou encore les avantages liés aux transmission MIMO (Multiple Input Multiple Output). Enfin, notons que l'outil proposé est validé dans ce travail sur des nœuds spécifiques, mais l'approche employée est générique et permet d'étendre l'outil à d'autres types de nœuds IoT.

II/ FIT IoT LAB : Description

II.1/ Une infrastructure publique et ouverte

La recherche en informatique française s'est dotée depuis plusieurs années de plateformes ouvertes permettant l'expérimentation et l'enseignement à distance. Ainsi dans le contexte de l'IoT, la plateforme FIT IoT Lab [1] est déployée sur plusieurs sites académiques en France. Cette plateforme est publique et ouverte à tous les membres des différents laboratoires de recherche français. Chaque site membre possède une plateforme matérielle propre sur laquelle les utilisateurs peuvent déployer les composants logiciels de leur choix afin de mener différentes expériences en IoT ou robotique. La plateforme peut aussi être utilisée à des fins d'enseignement de la programmation de microcontrôleurs et ou de robotique.

Une interface de programmation unifiée permet aux utilisateurs de la plateforme de réserver des ressources matérielles, de téléverser et exécuter leurs programmes sur les composants de la plateforme ainsi que de collecter des mesures sur diverses métriques telles que la consommation électrique des nœuds, par exemple.

Différents composants logiciels de la plateforme sont accessibles sur la plateforme Github de l'équipe FIT et distribués sous licence CECILL [2, 3].

II.2/ Architecture des nœuds FIT IoT

L'architecture FIT est composée d'un ensemble de nœuds que les utilisateurs peuvent réserver et re-programmer soit pour des expérimentations dans la recherche en réseau et / ou IoT, soit dans un but didactique. L'architecture logique d'un nœud IoT Lab est représentée schématiquement sur le diagramme en Figure 1, et comporte trois éléments :

Le Nœud Ouvert (Open Node, ON) est une carte programmable dotée d'un microcontrôleur et de multiples capteurs / actuateurs. C'est la carte cible vers laquelle un utilisateur de la plateforme peut envoyer un programme (firmware) pour exécution. Ces éléments peuvent être équipés de moyens de communication radio (wifi, lpwan, etc …)

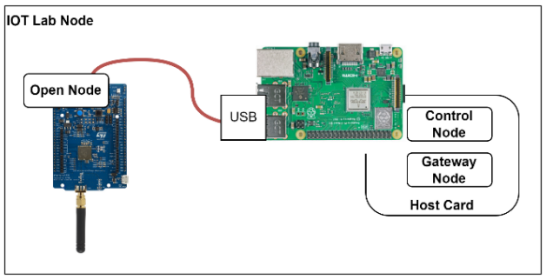


Figure 1: Représentation Schématique des composants matériels de la plateforme IoT Lab; repris de [1]

Chaque ON est connecté en USB a une carte IoT Lab Hôte [1]. La carte Hôte est constituée de deux éléments matériels appelés respectivement Gateway Node (GN), et Control Node (CN).

Le Nœud passerelle (Gateway Node, GN) est une carte dotée d'un CPU permettant d'exécuter un système linux. Ce système exécute une passerelle logicielle : l'iotlab-gateway [3] qui fournit une API REST permettant à un client de manipuler l'ON connecté à la carte hôte, notamment :

1. De redémarrer et reprogrammer l'ON avec un programme fournit par l'utilisateur.
2. De se connecter pour communiquer de manière directe avec l'ON via son port série.
3. Rétablir une configuration par défaut sur l'ON afin de pouvoir y téléverser de nouveaux programmes.

Cette passerelle sert donc de programmateur pour l'ON cible.

Le Control Node (CN) est un composant physique additionnel de la carte hôte doté de capteurs permettant de mesurer la consommation électrique des ON ainsi que la puissance du signal émit dans le cas des ONs équipés d'émetteurs. Le CN peut être déployé ou non selon les possibilités du matériel utilisé.

En fonction des différents sites de déploiement, le matériel de chaque nœud peut varier mais l'API de programmation des cartes reste la même.

Sur chaque site de déploiement, un serveur d'expérimentation permet aux utilisateurs de la plateforme de réserver plusieurs nœuds pour réaliser leurs tests.

L'architecture IoT LAB requiert le déploiement d'une carte Hôte pour chaque ON déployé. Ce choix a été fait pour garantir à la fois l'isolation des composants logiciels et les performances des mesures effectuées par le CN. Cependant le coût de déploiement pourrait être réduit en mutualisant le matériel des cartes hôtes. Nous décrivons une telle proposition dans la section suivante.

III/ Architecture proposée : le nœud Multi ON

Le nœud multi ON est une itération sur la base des nœuds de l'architecture FIT. Il permet de supporter l'exécution de plusieurs GN virtuels afin de permettre à une seule carte passerelle de gérer plusieurs ON. La Figure 2 représente la structure physique d'un nœud Multi ON.

III.1/ Partie matérielle

Physiquement le nœud multi ON, représenté en Figure 2, est composé d'un Gateway Host (GH), et de plusieurs Open Nodes. Le Gateway Host rempli la fonction de plusieurs Gateway Nodes. Pour cela, le Gateway Host exécute une instance virtuelle de Gateway Node pour chaque Open Node connecté à sa carte. Physiquement, le Gateway Host est une carte SoC dotée de plusieurs ports USB sur lesquels sont connectés plusieurs ON. Dans le prototype décrit ici, le rôle de GH est assuré par un Raspberry PI 3B, doté d'1GB de RAM et d'un CPU Broadcom cadencé à 1.2 Ghz. Au niveau connectique, le RPI3B est équipé d'un port Ethernet, d'une connexion wifi et de 4 ports USB. Ce composant a une capacité suffisante pour exécuter plusieurs instances de l'iotlab-gateway en parallèle. De plus, le RPI3B est capable de fournir les 1200 mA nécessaires à l'alimentation des périphériques USB.

Les ONs du prototype sont 4 cartes ST B-L072Z-LRWAN1, équipées d'un microcontrôleur basse consommation Arm Cortex v0+ et d'un module de communication LORA ainsi que de plusieurs GPIO. La puissance maximale utilisée par ces cartes d'après le constructeur est de l'ordre de 300 mA [4]. Ces cartes peuvent être connectées à plusieurs capteurs afin d'observer leur environnement.

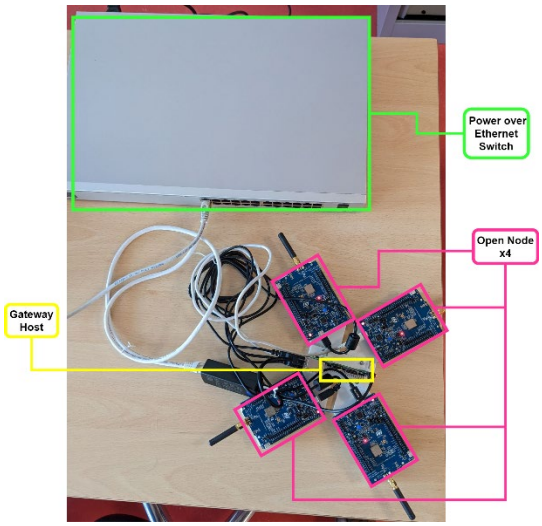


Figure 2: Photographie d'un nœud Multi ON : La carte Gateway Host (Raspberry PI 3b+) est alimentée par un switch PoE et permet l'alimentation de 4 Open Nodes (cartes ST B-L072Z-LRWAN1)

Un switch Power over Ethernet alimente les GN RPI, à travers un Dongle PoE to mini USB. Les ONs sont alimentés à travers la connexion USB au RPI et le hub USB du RPI est capable de supporter une charge de 1200mA maximum.

III.2/ Partie logicielle

Le rôle de la partie logicielle du nœud multi-ON est de permettre l'accès à l'interface de programmation et à la connexion série de chacune des cartes connectées au Gateway Host.

Pour cela, deux approches sont possibles : la première consiste à adapter le logiciel serveur de la Gateway IoTLAB [3] pour lui permettre de distinguer plusieurs ONs. En plus de l'effort de développement nécessaire, cette approche nécessiterait des changements dans la communication entre le serveur d'expérimentation IoT et le Gateway iot lab, le rendant incompatible avec les autres composants de la plateforme.

La seconde approche, que nous avons adoptée dans ce travail, consiste à exécuter plusieurs instances du serveur iot-ab-gateway sur un seul et même GH. La technologie de conteneurisation Docker [4] a été utilisée pour exécuter plusieurs instances de passerelles (une pour chaque ON). Le composant iotlab-gateway est peu gourmand en ressources, ce qui compense l'augmentation de la charge due à l'exécution de plusieurs instances avec Docker.

Les cartes ST utilisées sont dotées d'une interface ST-Link V2.1 permettant la communication et la reprogrammation du microcontrôleur via une connexion USB.

III.2.1/ Conteneurisation de l'iotlab-gateway

La conteneurisation est une méthode de déploiement d'applications qui permet de regrouper un programme et ses dépendances dans une image système facilement transférable entre différents hôtes. Un exécuteur de conteneurs tel que Docker [5] permet l'exécution de cette image dans un processus isolé du reste du système d'exploitation. Le mécanisme d'isolation repose sur la restriction des permissions du processus exécutant le conteneur. La conteneurisation est une approche moins gourmande en ressources que la virtualisation car au contraire d'une machine virtuelle, qui inclue une image système entière, un conteneur partage le noyau du système d'exploitation avec son hôte.

Le projet iotlab-gateway fournit déjà les sources nécessaires à la construction d'un conteneur pour permettre différents tests du code de la passerelle. La construction du conteneur pour notre prototype a donc consisté à alléger le conteneur en enlevant les dépendances inutiles pour la gestion des cartes de notre prototype.

III.2.2/ Association ON-Conteneur

Un conteneur n'est pas autorisé à voir ou accéder les périphériques connectés à son hôte. Pour autoriser un conteneur à accéder aux périphériques de l'hôte, deux approches sont possibles : l'exécution en mode privilégié ou l'autorisation par périphérique.

L'exécution en mode privilégié donne toutes les permissions du système hôte au conteneur. Bien que cela permette le pilotage des périphériques, cette approche présente deux inconvénients : premièrement cela présente un risque de sécurité important car le conteneur aurait les mêmes privilèges qu'un utilisateur root sur l'hôte. Le second inconvénient de cette approche est que tous les périphériques deviendraient visibles à tous les conteneurs, or le service iotlab-gateway a été conçu avec l'hypothèse de départ qu'une passerelle n'est connectée qu'à un seul ON.

Pour ne pas utiliser le mode privilégié, il est possible d'utiliser l'autorisation par périphérique. Dans cette approche, le conteneur ne s'exécute pas en mode privilégié : la liste des périphériques auquel le conteneur a accès est fournie explicitement au lancement du conteneur. Il est ainsi possible de garantir que chaque conteneur ne voie qu'un seul ON. C'est l'approche que nous utilisons pour le déploiement de la passerelle multi-ON.

Dans le cas des cartes ST utilisées pour le prototype, il est nécessaire d'autoriser deux périphériques pour chaque carte : l'interface de communication modem ACM (Abstract Control Model) et le bus USB du périphérique. Le script de lancement utilise le paramètre `--device` pour associer ces deux périphériques au

démarrage du conteneur. Un exemple de commande de lancement pour un conteneur est reproduit en figure 3.

```
docker run --rm --name=iot-labisen-0 --net=gw_net \
 -h iot-labisen-0 --ip=192.168.0.10  \
 --device /dev/ttyACM0:/dev/iotlab/ttyON_STLINK  \
 --device /dev/bus/usb/001/003:/dev/bus/usb/001/003 \
 -d iotlab-gateway:1
```

Figure 3 : Commande de lancement d'un conteneur iotlab-gateway

Pour assigner le bon périphérique à un conteneur il est nécessaire d'identifier le numéro de bus USB et le périphérique série correspondant. Le numéro de bus USB est susceptible de changer en cas de déconnexion / reconnexion physique de la carte. En revanche, le numéro de Port USB sur lequel la carte est connectée ne change pas, peu importe le nombre de connexions ou déconnexions de la carte. Ce numéro de port est récupérable à l'aide de la commande `lsusb` sous linux.

Un script bash a été écrit pour permettre le traitement de la commande `lsusb` et d'en déduire l'association port USB / numéro de bus USB. Cette information est ensuite utilisée par un script de lancement au démarrage du RPI3B, qui énumère les périphériques USB connectés au GH et lance un conteneur de passerelle pour chaque interface ON reconnue. Ce script de lancement attribue une adresse IP, un nom d'hôte et un ON à chaque conteneur.

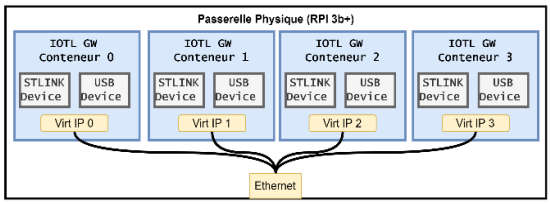


Figure 3: : Vue Schématique des composants logiciels du nœud Multi ON

III.2.3/ Connectivité des passerelles

Les passerelles iotlab-gateway reçoivent des commandes et instancient les communications séries sur des ports TCP prédéfinis. Or le RPI3B n'a qu'une seule interface Ethernet physique. Cette interface est donc partagée avec la technologie de virtualisation d'interface réseau IP Vlan qui permet d'assigner plusieurs adresses IP à une même adresse MAC (cf. Figure 4). L'association interface virtuelle – conteneur est assurée par le gestionnaire de conteneurs. Cette approche permet à chaque conteneur d'utiliser les mêmes numéros de ports. Ainsi chaque passerelle est vue comme un nœud réseau unique par le serveur d'expérimentation ou les clients. Aucun changement n'est donc requis pour ces composants.

III.3/ Evaluation de la charge du GW host

Nous avons évalué l'impact de l'exécution de 1 à 3 conteneurs iotlab-gateway sur un RPI3B en mesurant la température CPU, l'occupation CPU ainsi que l'occupation mémoire. Le test consiste à observer les métriques durant le lancement des conteneurs, avec une pause de 5 secondes entre chaque lancement.

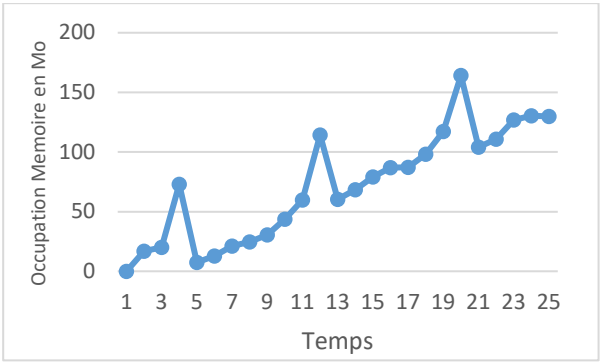


Figure 4: Occupation mémoire due à l'instanciation des conteneurs

Le Gateway Host est un Raspberry 3b-équipé d'un système Ubuntu linux 20.04 avec le noyau 5.4.0-1059-raspi pour arm64.

Les mesures sont effectuées avec l'outil vmstat pour les statistiques CPU et Mémoire, et l'outil « sensors » pour la température CPU.

La figure 5 montre l'augmentation de la consommation mémoire due à l'instanciation des conteneurs ; on distingue 3 pics liés à l'instanciation de chaque conteneur. À la suite de l'initialisation la consommation se stabilise avant l'apparition d'un nouveau conteneur. A partir du premier pic, on peut estimer qu'environ 75Mo sont nécessaires pour l'instanciation d'un conteneur. Une telle consommation est acceptable pour une plateforme telle que le RPI qui est équipé avec 1Go de RAM. A l'utilisation on constate qu'une instance de passerelle nécessite environ 50Mo de mémoire vive.

La Figure 6 montre la variation de la température et de l'utilisation CPU du RPI au cours du démarrage des conteneurs. Là encore on peut observer que les ressources disponibles sur le RPI sont suffisantes pour supporter la charge des conteneurs. L'utilisation CPU ne dépasse pas 50% et la température reste dans un intervalle acceptable pour une utilisation normale du RPI (en dessous de 80°C).

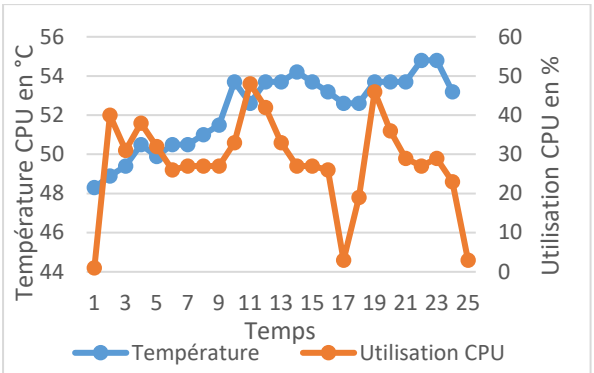


Figure 5: Température et Utilisation CPU pendant le lancement des conteneurs

IV/ Application à l'hôpital connecté

Une maquette d'hôpital connecté a été réalisée par des élèves de M1. Dotée de multiples capteurs et actuateurs, cette réalisation fournit un environnement propice au déploiement de multiples applications soit pour des projets d'élèves soit dans le cadre de la recherche en IoT.

La maquette d'hôpital connecté (figure 7) est constituée de plusieurs pièces, chacune dotées d'un ensemble de capteurs et d'actuateurs spécifiques à leur fonction. Par exemple la salle d'examen (1) sur la maquette est dotée d'un cardio-fréquence mètre. La salle de réserve (3) est dotée d'un capteur d'empreinte digitale, et d'une porte actionnable tous deux connectés à un même microcontrôleur permettant d'implémenter une boucle de contrôle d'accès (2).

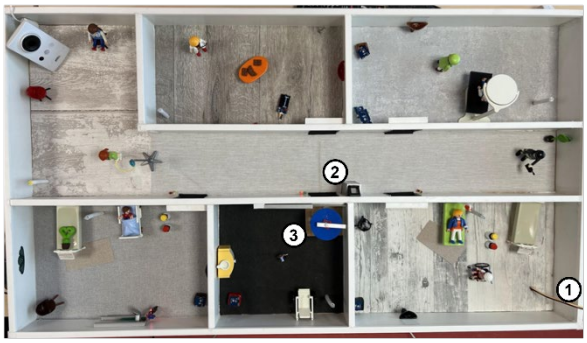


Figure 6: Maquette d'hopital connecté

Ces différents capteurs et actuateurs sont pilotés soit par microcontrôleurs de type ST LRWAN ou Arduino, qui étaient programmés manuellement. Le déploiement d'un nœud Multi ON permet le contrôle et la reprogrammation à distance des différents éléments actifs de la maquette.

V/ Conclusions et travaux futurs

Ce travail fournit les bases d'une plateforme conteneurisée pour la gestion des nœuds d'un système compatible avec la plateforme FIT IoT LAB. Nous démontrons qu'il est possible de mutualiser certains composants matériels de la plateforme grâce à la technologie de conteneurisation Docker, tout en conservant une compatibilité fonctionnelle avec les autres composants logiciels du système.

Dans le futur, nous nous attacherons à concevoir un nœud de contrôle adapté au nœud multi ON afin de pouvoir observer la consommation électrique et les émissions radio des ON. En parallèle, l'étude de différentes technologies de conteneurisation permettra d'identifier la solution avec la consommation de ressources la plus réduite.

VI/ Bibliographie

[1] C. Adjih *et al.*, « FIT IoT-LAB: A large scale open experimental IoT testbed », in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, Italy, déc. 2015, p. 459-464. doi: 10.1109/WF-IoT.2015.7389098.

[2] CeCILL [En ligne], Consulté le : 15 juin 2022 Disponible sur : http://www.cecill.info/

[3] *IoT-LAB Gateway*. Consulté le: 16 juin 2022. [En ligne]. Disponible sur: https://github.com/iot-lab/iot-lab-gateway

[4] « Discovery kit for LoRaWAN™, Sigfox™, and LPWAN protocols with STM32L0 ». 30 juin 2018. [En ligne]. Disponible sur: https://www.st.com/resource/en/user_manual/um2115-discovery-kit-for-lorawan-sigfox-and-lpwan-protocols-with-stm32l0-stmicroelectronics.pdf

[5] D. Merkel, « Docker: lightweight linux containers for consistent development and deployment », *Linux Journal*, vol. 2014, nº 239, p. 2, 2014.

CHALLENGE EXTREME DEFI « VEHICULE FAIBLE IMPACT » (ADEME)

A. SIVERT [1] ,B. VACOSSIN [1], F. BETIN [1], G. PLASSAT [2]
(1) I.U.T de l'Aisne Département Génie Électrique SOISSONS FRANCE
Laboratory for Innovative Technologies (L.T.I), Team Energy Electric and Associated System
(2) ADEME

Résumé : *De nombreuses questions se posent en ce qui concerne le choix des véhicules électriques en fonction des besoins des utilisateurs, des impacts énergétiques et planétaire. Parmi le choix de véhicules proposés, des "concept car" voient le jour pour un budget et une consommation d'énergie très variés. Dans ce cadre, l'Agence de la Transition Écologique (ADEME) lance un défi pour la réalisation de véhicules pour lesquels il serait envisageable de baisser les coûts de production et de limiter les consommations énergétiques tout en minimisant leurs impacts sur l'environnement donc des véhicules durables, équitables et viables..*

I/ Introduction

La transition énergétique d'une part et les évolutions géopolitiques d'autre part engendrent des contraintes de plus en plus fortes sur les budgets des ménages et entrainent de nécessaires mutations sociales. Dans le cas de la mobilité, l'optimisation du choix du type de véhicules (voiture, moto, scooter, vélo, trottinette, …) en fonction des besoins prend une importance de plus en plus grande. En effet, jusqu'à présent, l'optimisation du facteur de mobilité conduisait en général à mettre en avant la satisfaction du besoin sans tenir compte des aspects écologiques car l'impact sur les budgets restait faible (environ 4% pour la classe moyenne à 17% pour les ménages modestes) mais les choses changent...

En complément du choix du véhicule, l'intermodalité apparaît comme une réponse pertinente à ce besoin d'optimisation. Elle est déjà une réalité pour de nombreuses personnes qui cherchent à réduire leurs impacts sur la planète mais aussi à faire preuve de bon sens ou profiter du plaisir de piloter ces nouveaux véhicules. Toutefois des contraintes spécifiques comme la nécessité de prendre une assurance pour chaque véhicule sont à prendre en compte. Il faut donc ici aussi trouver des solutions innovantes. Une assurance basée sur le véhicule le plus coûteux et présentant le facteur de risques le plus élevé mais qui ouvrerait l'ensemble des véhicules utilisés permettrait de contenir la charge financière des assurances.

Enfin, chaque territoire possède des caractéristiques propres ce qui implique des mobilités différentes en fonction de l'éloignement des services, du dénivelé, des transports collectifs existants, …

La demande de nouveaux véhicules existe et tous les ans, pléthore de nouveaux engins voient le jour. La variété de ces productions est telle que la mise en place d'outils de comparaison n'est pas aisée [1].

En règle générale ces nouveaux prototypes sont :

- soit trop en avance par rapport à la demande,
- soit trop décalés par rapport à la concurrence des véhicules classiques et échouent à devenir des produits de masse. Leurs coûts restent alors trop élevés. On peut citer « Evovelo » qui a jeté l'éponge en 2021 au bout de 8 ans de développement et production [2].

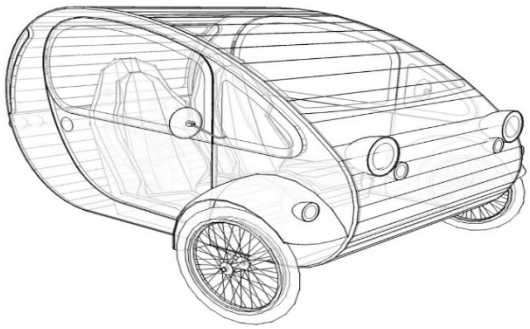


Figure 1 : Montage bois et plexiglass Evovelo

Ainsi, de nombreux ingénieurs et bricoleurs passionnés s'intéressent à la réalisation de prototypes. Ils réalisent des « Objets Roulants Non Identifiés » (Unidentified Rolling Object) en cherchant à répondre au mieux à leurs propres besoins de mobilité, malgré les barrières des règlementations, homologations, assurance.

L'objectif de « l'extrême défi » décrit dans cet article est de créer une catégorie de véhicules sobres, efficaces, peu coûteux, interopérables au sein d'un écosystème local cohérent, simple à assembler, utilisant des composants standards de masse et recyclables et permettant d'optimiser chaque usage**.**

Objectif de l'extrême défi

L'objectif est de concevoir et produire un véhicule « 300 fois plus intéressant » que la voiture tout en permettant de transporter deux personnes et une charge

de 100 kg en utilisant des matériaux minimisant l'impact sur la planète (recyclable à l'infini).

Ce coefficient de 300 correspond à :

- Réduire le coût énergétique par 10 par rapport à celui d'une voiture qui est d'environ de 0,6 à 1 €/km.
- Augmenter d'un facteur trois la durabilité. L'âge moyen des voitures est actuellement de 11 ans.
- Réduire d'un facteur 10 la consommation d'énergie. Une voiture moyenne consomme environ 150 Wh/km à 45km/h.

Pour atteindre cet objectif, la masse du châssis du véhicule doit être 10 fois plus faible que celui d'une voiture moyenne (1300 kg). La vitesse maximale devra être inférieure à 65 km/h avec des technologies simples et abordables (« low tech ») pour atteindre une maintenabilité aisée et donc durable. Ces véhicules sont communément nommés « véhicules intermédiaires » [3].

Le défi consiste à concevoir puis produire son prototype en 3 ans :

- Première année : une phase d'idées, de choix, d'analyse pour réaliser un cahier des charges et un dessin mécanique.
- Deuxième année : une phase de réalisation d'un prototype.
- Troisième année : démarche de petite production, incluant l'industrialisation et la prise en compte des aspects logistiques [4].

A ce jour, **273 candidats** formant **43 équipes** répondent à la première phase de la saison 2022 du chalenge. La gestion de projet fait appel à des ressources gérées par l'intermédiaire d'un wiki et associée à un vocabulaire spécifique.

Les différents acteurs sont :

- **155 « Compagnons »** qui peuvent aider les équipes dans leurs choix et méthodes (Ingénieurs, designers, experts en modèles économiques ou en changement, chercheurs en économie circulaire, experts en normes et certifications, …)
- **158 « Témoins »** qui sont déjà des utilisateurs (associations militantes, curieux, fans…). Par ailleurs, 20 territoires se préparent à l'extrême défi (XD) en prévoyant d'accueillir des solutions de remplacement à la voiture.
- **32 « Partenaires »**, qui apportent des ressources (matériels, territoires, financements, entreprises, ...) et identifient des compagnons issus de leurs organisations.
- Des groupes de travails (GT) qui se sont auto structurés (batterie [7], business [8], ...)

Ainsi, la conduite de projet du challenge fait appel à un wiki et à un forum qui accompagnent une création participative et discursive. Dans ce cadre, les « candidats », « les témoins », « les acteurs » doivent s'inscrire et peuvent créer des ressources.

De plus, il est recommandé de travailler avec des logiciels open source. Toutefois, les experts qui ont l'habitude d'utiliser des logiciels payants, en général plus performants, ne seront pas discriminés.

Les équipes comportant au minimum 3 personnes ont dû mettre leurs cahiers des charges et leurs objectifs en ligne.

La phase d'enregistrements des équipes est terminée. Elle s'est accompagnée de la définition de leur choix de cahier des charges. Les équipes doivent remplir des dossiers incluant le narratif, l'éco système, l'aspect économique, le retour d'expériences, [5] ainsi que la présentation du véhicule et l'aspect énergétique du projet. [6]

Il est cependant toujours possible d'être « acteur » et « compagnon » : https ://wiki.lafabriquedesmobilites.fr/wiki/Devenir_acteur_de_l'XD

La pluridisciplinarité intrinsèque au projet nécessite beaucoup de compétences (mécaniques, matériaux, motricité, ergonomie, aérodynamisme, freinage, éclairage, pneu, système électrique, …). La réalisation d'un prototype opérationnel est alors un excellent support pédagogique collaboratif pour les écoles en faisant intervenir différentes filières.

Pour prendre la mesure de ce défi, commençons par rappeler la situation actuelle. Ainsi : quelles sont les performances énergétiques des véhicules à ce jour ?

II/ Efficacité énergétique de quelques véhicules

Le tableau 1 suivant donne un ordre d'idée des performances de différentes familles de véhicules. Le prix dépend de la capacité énergétique des batteries, du nombre de ventes, de la motorisation, des marges constructeurs, des matériaux utilisés, du marketing …

Par exemple, les vélos cargos ont souvent des motorisations « premium » (marque haut de gamme) avec des batteries qui coûtent de l'ordre de 1,6 €/Wh alors que le prix classique d'une batterie est d'environ 0,3 €/Wh. Pourtant, les cellules des batteries sont identiques. Notons aussi que la maintenabilité est meilleure pour les batteries et les moteurs dans le cas des vélos non-premiums car ils ne sont pas incorporés dans le cadre et peuvent être changés facilement.

Dans le tableau suivant le « taux de poids mort » correspondant à la masse à vide par rapport à la masse en charge maximale autorisée est très faible pour les petits véhicules ce qui permet de minimiser la consommation énergétique.

Les données du tableau 1 montrent que la masse du véhicule augmente en fonction de la charge transportée et qu'elle impacte ainsi fortement l'efficacité énergétique. On comprend donc l'intérêt de la recherche sur les véhicules intermédiaires dont le poids mort doit être inférieur à 50% [19]. Mais la masse d'un véhicule

est aussi liée à sa vitesse maximale car elle est associée à une sécurité passive acceptable. On se rend bien compte qu'un vélo en descente peut rouler à plus de 90km/h, mais avec peu de sécurité en cas de sortie de route ! L'acceptabilité sociale des risques demandent des procédures, des expertises, des normes, des réglementations et porte souvent à des controverses.

Caractéristiques	Masse du véhicule (kg)	Puissance maximale du moteur (W)	Vitesse moyenne (km/h)	Puissance utilisée à la vitesse moyenne (W)	Efficacité énergétique (Wh/km)	Volume (litre) et masse de transport	Taux poids mort (%)	Budget moyen en 2022
Calcul pour :			b	a	a / b			
Vélo mobile	30	-	35	100	3	70l/120kg	20	9 000€
Vélo	14	-	18	100	5,5	70l/100kg	12	1 000€
Trottinette électrique	14	250	20	250	12,5	15l/100kg	12	350€
Vélomobile à assistance électrique	50	250	45	350	8	70l/150kg	25	10 000€
Vélo à assistance électrique	20	250	25	250	10	70l/130kg	13	2 000€
Vélo cargo électrique et speed pedelec	40	1500	45	1400	31	140l/200kg	16	5 500€
Scooter électrique	90	4 500	45	1500	33	40l/200kg	31	3 000€
« Voiturette » électrique	450	6 000	45	3 400	77	140l/240kg	65	7 000€
Petite voiture électrique	920	33 000	45	5 800	130	500l/300kg	75	16 000€
Voiture électrique	1 500	80 000	45	6 700	150	600l/400kg	78	35 000€

Table 1 : Efficacité énergétique de quelques véhicules

Pour minimiser l'impact sur la planète, un véhicule avec des matériaux biosourcés serait intéressant à concevoir, mais en général, ces matériaux n'optimisent pas la masse, ni le budget. Notons par ailleurs, que pratiquement tous les matériaux peuvent être ressourçables et recyclables à condition que la filière existe et que le tri soit efficace.

Pour allonger le cycle de vie des produits et minimiser les coûts, l'«extrême défi» stipule qu'il est nécessaire de mettre en place une interopérabilité des produits.

III/ L'interopérabilité, compatibilité, norme, maintenabilité, logistique

L'interopérabilité est la capacité à rendre compatibles deux systèmes quelconques. Elle nécessite que les informations nécessaires à sa mise en œuvre soient disponibles sous la forme de « standards ouverts ».

L'interopérabilité d'un véhicule électrique devra par exemple conduire à avoir le même protocole de communication pour une même instrumentation qui peut se retrouver aussi bien sur une console filaire, une console Bluetooth, un smartphone, un chargeur Bluetooth, ...

Autre exemple, si les possesseurs de vélos électriques possédaient deux batteries en parallèles (l'une en location et l'autre personnelle), elles devraient être de mêmes dimensions avec les mêmes prises de charge et de décharge pour être interchangeables. Cela faciliterait l'autonomie de déplacement en permettant un échange des batteries en quelques minutes. Il serait possible aussi, de mettre un certain nombre de ces batteries en parallèles pour être compatible avec des véhicules plus puissants comme les scooters ou les voiturettes.

On peut noter que certains fabricants d'outillages électroportatifs utilisent les mêmes batteries pour différents outils, minimisant ainsi le coût des différents systèmes.

Par ailleurs, les standards sont souvent très en retard sur les solutions de recharge commercialisées. Il n'est pas rare d'attendre plus de 10 ans pour qu'un standard existe et s'impose pour devenir une norme. Ainsi, à ce jour, il n'existe pas de standard pour les vélos électriques pour les batteries ou les recharges.

Autres points qui limitent l'interopérabilité :

- L'ensemble « cadre + batterie » est souvent interdépendant alors lorsque la batterie est défectueuse tout le système doit être recyclé (le retrofit des batteries n'est pas rentable du fait de la durée des opérations de montage/démontage).
- Les systèmes premiums ne sont pas « ouverts » pour éviter un débridage qui sont assez facilement réalisables en électrique (vitesse puissance).
- Sur de nombreux véhicules premiums, il est très difficile d'avoir la possibilité d'obtenir des pièces et de faire les réparations soi-même. Cette limitation de la maintenabilité est d'ailleurs souvent pointée du doigt dans les magazines de consommateurs [9].

Dans le cadre d'une petite production, la qualité et la disponibilité des différents composants est un problème important qui n'a fait que de s'accentuer ces dernières années. La disponibilité et le prix des composants dépendent principalement du volume acheté à l'année et du pays de production. Au sein des départements de génie électrique des IUT, la question de la logistique et du remplacement des composants est peu étudiée mais ce sont des éléments importants pour les départements QLIO (Qualité, Logistique Industrielle et Organisation).

L'impact d'un produit sur le prix et l'empreinte carbone dépend aussi du trajet depuis le lieu de production. La question de la relocalisation de la production revient donc au-devant de la scène. A petite échelle, pour les besoins des écoles et des start-ups, il existe des sites de vente unitaire de composants permettant d'assembler des prototypes de véhicules intermédiaires [11].

Les choix de technologies sont multiples :

- Traction hybridée ou non, musculaire, série parallèle [12].
- Nombre de roues pour favoriser la stabilité ou pour la capacité à se faufiler dans le trafic.
- Système pendulaire ou non,
- Nombre de roues motrices…

Les solutions envisageables sont alors d'une grande variété. Mais le pragmatisme, l'optimisation du budget, la maintenabilité limitent rapidement les possibilités.

Pour assurer une bonne maintenabilité, il est souhaitable que le produit soit « open source ». Cela permet de disposer d'une grande communauté d'entraide et de facilement modifier les caractéristiques en fonction de l'utilisation. Ainsi, les systèmes « open source » favorisent la collaboration et le partage. Notons à ce propos qu'il faut que tous les plans, les pièces, les composants soit connus ce qui demandent d'avoir des outils participatifs efficaces et une bonne méthodologie.

IV/ Véhicules Open Source

Les véhicules industriels sont « fermés » et ne permettent pas une grande maintenabilité à l'inverse des véhicules « open source » dont la conception plus ouverte permet de « débrider » les caractéristiques techniques.

Par ailleurs on pourrait croire que les petits véhicules comportant peu de pièces et pouvant être fabriqués soi-même sont bon marché. Il n'en est rien. En effet, à titre d'exemple, le « peel-p50 » qui est un véhicule des années 60 et dont la production a repris en 2018 [17, 18], ou le vélo pliant Brampton qui est considéré comme la roll Royce du vélo, désuet et d'un autre âge a un coût exorbitant par rapport à tous ses concurrents.

A contrario, il existe déjà plusieurs projets de véhicules intermédiaires « open source » comme Vhélio et Mosquito.

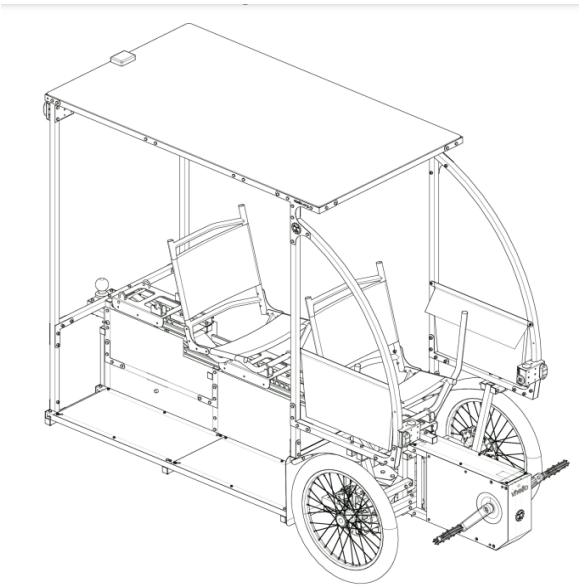


Figure 2 : Montage aluminium Vhélio

Des sous-projets accessibles à des étudiants en Génie Electrique peuvent être aussi intéressants à mener. Par exemple, la réalisation et le paramétrage d'un contrôleur en fonction de son besoin [15]. Il faut toutefois respecter les limites légales comme la vitesse maximale du véhicule qui pourrait être dépassée par un nouveau paramétrage [16].

La réalisation d'un véhicule entièrement « open source » nécessite aussi une réflexion approfondie sur la méthode d'organisation des données pour que la communauté puisse profiter efficacement des différentes réalisations [13]. La lecture des fichiers doit être, elle aussi, « open source ».

De nombreux projets mal renseignés sur le net restent à l'état de projet et ne sont jamais concrétisés car le défaut d'organisation complique le partage d'informations. Certaines expériences intéressantes dont les données en sont restées au « papier-crayon » sont elles aussi inaccessibles à la communauté et restent-elles aussi à un niveau embryonnaire.

Dans le cadre du challenge présenté dans cet article, la communauté est active et tous les jeudis matin entre 9h et 10h, des réunions en distanciel sur Zoom sont organisées. Des comptes rendus sont disponibles pour ceux qui n'ont pas pu être présents [14].

V/ Conclusions

L'« extrême défi » est un challenge aux multiples facettes (participatifs, techniques, économiques, éthiques, …). En effet, pour optimiser les déplacements, le choix d'un véhicule devrait se faire comme celui d'une paire de chaussures, en fonction du besoin et de la météo. Le changement des mentalités sur les véhicules faibles consommations de tous les jours, restera toujours aussi limité si des progrès importants ne sont pas faits aux niveaux des coûts et de la maintenabilité.

L'ADEME lance donc un défi ambitieux en visant une réalisation à bas coûts de véhicules à faibles

consommations et pour pouvoir disposer d'une multitude de prototypes afin de mieux répondre aux besoins des utilisateurs.

Des questions restent en suspens :

- Les réglementations et les homologations (européennes et françaises) permettront-elles de promouvoir ces nouveaux véhicules intermédiaires ?
- L'espace public pourra-t-il être utilisé par ces véhicules intermédiaires ?
- Les mentalités des utilisateurs évolueront elle suffisamment favorablement pour que ces véhicules trouvent leurs places ?
- Les « business model » des différents acteurs permettront-ils l'essor de ces productions tout en minimisant l'impact sur la planète ?

VI/ Bibliographie

[1] A.Sivert, B.Vacossin, S.Carriere, F.Betin « Etude technique comparative de différents véhicules électriques de 50 kg à 1500 kg vitesse, consommation batteries, prix, modes de charge » , Revue 3EI octobre 2021

[2] https://www.evovelo.com/en/index.php#team

[3] https://xd.ademe.fr/blog/vous-avez-dit-vehicule-intermediaire-comme-c-est-bizarre

[4] https://xd.ademe.fr/

[5] https://wiki.lafabriquedesmobilites.fr/wiki/Equipe_HPULHV_(Human_powered_Ultra_light_hybrid_vehicle)

[6] Véhicule HPULHV (Human powered Ultra light hybrid vehicle) — Communauté de la Fabrique des Mobilités (lafabriquedesmobilites.fr)

[7] _Echanges du GT Batterie XD — Communauté de la Fabrique des Mobilités (lafabriquedesmobilites.fr)

[8] https://wiki.lafabriquedesmobilites.fr/wiki/Echanges_du_GT_March%C3%A9_Business_mod%C3%A8le_XD

[9] https://www.60millions-mag.com/2022/04/28/electrifier-son-velo-oui-mais-gare-la-securite-19956

[10] https://www.banquedesterritoires.fr/le-programme-territoire-engage-transition-ecologique-de-lademe-gagne-petit-petit-du-terrain

[11] https://www.kit-elec-shop.com/fr/

[12] https://hal.archives-ouvertes.fr/tel-03120708/file/THESE_DE_DOCTORAT_TOURNON_EDGAR_ONLINE.pdf

[13] https://wiki.lafabriquedesmobilites.fr/wiki/Ateliers_des_outils_pour_l%27XD

https://lafabriquedesmobilites.fr/blog/changer-mobilites-implique-de-nouvelles-facons-de-travailler-ensemble

[14] https://wiki.lafabriquedesmobilites.fr/wiki/Cat%C3%A9gorie:CR_Atelier

[15] _https://opensourceebikefirmware.bitbucket.io/

[16] _https://toutdebrider.fr/kit/debrider-velo-electrique-bosch/

[17] https://creapills.com/peel-p50-voiture-kit-20220623

[18] https://en.wikipedia.org/wiki/Peel_P50

[19] Revue transport urbains « véhicules intermédiaires » n 141 septembre 2022

Figure 3 : Vélomobile 50kg, autonomie 200 km à 45km/h

Figure 4 : Vélocouché (35kg, 3KW, 200km autonomie à 45km/h)

Figure 5 : Objet Roulant Non Identifié

Anthony Juton, Xavier Rain, Valérie Sauvant-Moynot,
François Orsini, Christelle Saber, Seddik Bacha,
Olivier Béthoux, Éric Labouré

TECHNOLOGIES DES VOITURES ÉLECTRIQUES

Motorisations, batteries, hydrogène, recharge et interactions réseau

Préfaces de Patrick Bastard, directeur de la recherche, Groupe Renault et Carla Gohin, vice-présidente Recherche et Innovation, Stellantis

DUNOD

COLLECTION TECHNIQUE & INGÉNIERIE
ÉLECTROTECHNIQUE

ÉLECTRONIQUE | ÉNERGIES | FROID ET GÉNIE CLIMATIQUE | GESTION INDUSTRIELLE | ENVIRONNEMENT | MÉCANIQUE ET MATÉRIAUX | ÉLECTRO-TECHNIQUE | AUTOMATIQUE ET RÉSEAUX

TECHNOLOGIES DES VOITURES ÉLECTRIQUES

L'électrification des véhicules routiers est au cœur de la communication, des innovations et des investissements des constructeurs automobiles. Rarement l'histoire aura vu une filière industrielle effectuer une mutation aussi rapide, renonçant à une prédominance de la motorisation thermique au profit de la motorisation électrique.

Fruit du travail d'une équipe complémentaire de chercheurs, enseignants et ingénieurs, cet ouvrage détaille l'ensemble de l'écosystème impacté par l'électrification des véhicules :

- Les technologies présentes et à venir de la **motorisation** des véhicules électriques.
- Les **batteries** Li-Ion et les recherches pour les améliorer.
- Les **chargeurs** du quotidien comme les chargeurs très haute puissance.
- Les contraintes mais aussi les bénéfices liés aux **interactions des véhicules avec le réseau** électrique (V2G).
- Le fonctionnement et les avancées dans le domaine des piles à combustible pour l'usage de l'**hydrogène sur les véhicules routiers**.

En outre, deux annexes rappellent les fondamentaux des machines électriques et de l'électronique de puissance.

Ce livre accompagnera les ingénieurs et techniciens du secteur automobile dans cette nécessaire mutation des compétences, tout en évoquant les innovations à venir.

POINTS FORTS

- ✔ Une équipe d'auteurs associant enseignants/chercheurs et experts de l'industrie.
- ✔ Un champ de compétences très large, de l'électrochimie à la régulation du réseau.
- ✔ Une présentation complète et approfondie de l'écosystème voiture électrique, réseau et hydrogène compris.
- ✔ Un contenu à jour des innovations les plus récentes.
- ✔ Les perspectives pour le futur.

LES + EN LIGNE
Des compléments à l'ouvrage sont disponibles en téléchargement sur le site dunod.com.

9 782100 818068
8574515
ISBN 978-2-10-081806-8

Anthony Juton
Professeur agrégé de physique appliquée ENS Paris-Saclay.
Xavier Rain
Docteur et professeur agrégé de génie électrique IUT de Cachan.
Valérie Sauvant-Moynot
Chef de département IFPEN Électrochimie et Matériaux.
François Orsini
Chef de projet Innovation Renault.
Christelle Saber
Docteur/ingénieur EnPu Safran.
Seddik Bacha
Professeur des Universités Université Grenoble Alpes.
Olivier Béthoux
Professeur des Universités GeePs/Sorbonne-Université.
Éric Labouré
Professeur des Universités Université Paris-Saclay.

Ont également contribué à cet ouvrage :

Mimoun Askeur
Responsable R&D/Expert EnPu Valeo.
Larbi Bendani
Expert EnPu Valeo-Siemens.
Antoine Cizeron
Doctorant GeePs – SATIE.
Éric Gimet
Expert EnPu Stellantis.
Fabrice Le Berr
Chef de département IFPEN.
Régis Le Drézen
Chef de département Mobilité électrique Enedis.
Cédric Léonard
Chef de pôle RTE.
Sid-Ali Randi
Ingénieur R&D Renault/Vedecom.
Najib Rouhana
Docteur/ingénieur EnPu Safran.
Damien-Pierre Sainflou
Chef de projet Recharge Stellantis.

DUNOD
une page d'avance

ENVOYEZ VOS CANDIDATURES

GRANDS PRIX SEE 2022

Comme chaque année, la SEE lance les appels à candidatures pour ces Prix et Médailles de grand renom, afin d'identifier les personnalités scientifiques et techniques

GRAND PRIX DE L'ÉLECTRONIQUE GÉNÉRAL FERRIÉ | 65ÈME ÉDITION

Depuis 1963, le Grand Prix de l'Électronique « Général Ferrié » est décerné annuellement.
Il récompense un ingénieur ou scientifique dont les travaux ont contribué d'une manière importante aux progrès des Systèmes d'Information et de Communication, y compris dans leurs aspects énergétiques.

› Date limite de candidature : 30 septembre 2022

PRIX BRILLOUIN | 15ÈME ÉDITION

Ce prix est remis les années paires, en alternance avec le prix Glavieux qui est lui, remis les années impaires.

Les domaines concernés par le Prix Brillouin sont :
• physique des matériaux et des composants, optique, électronique et électrotechnique.

› Date limite de candidature : 9 septembre 2022

MÉDAILLE BLONDEL | 80ÈME ÉDITION

La médaille Blondel couronne chaque année des scientifiques universitaires ou industriels, français ou étrangers, pour des travaux remarquables contribuant aux progrès de la Science et des Industries Electrique et Electronique, et menés avec les mêmes soucis d'approfondissement et de rigueur que ceux qui caractérisaient les travaux d'André Blondel.

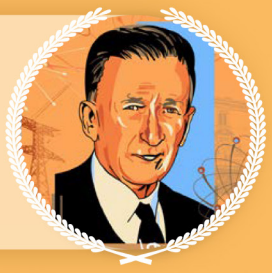
› Date limite de candidature : 9 septembre 2022

PRIX «JEUNES» ANDRÉ BLANC-LAPIERRE | 20ÈME ÉDITION

Les Prix «Jeunes» André Blanc-Lapierre ont été institués en mémoire de ce grand scientifique français, déclinés en Prix Régionaux puis en Prix National.

Ces distinctions s'adressent aux étudiants au niveau master ingénieur, des établissements français d'enseignement supérieur dans les domaines de l'électricité, de l'électronique, des technologies de l'information et de la communication, de se distinguer en bénéficiant d'une reconnaissance nationale.

› Date limite de candidature : 14 octobre 2022



POUR TOUT RENSEIGNEMENT :
www.see.asso.fr

Abonnez-vous à la REE

REVUE DE L'ÉLECTRICITÉ ET DE L'ÉLECTRONIQUE

La REE est une publication trimestrielle de la SEE

Choisissez votre formule d'abonnement pour 2022 :

❑ Version papier

5 numéros : mars, mai, juillet, octobre, décembre.
Distribution postale

France & UE	Hors UE
❑ 125 € TTC	❑ 142,43 € HT*

❑ Version numérique

Accès aux publications numériques
ouvert pendant un an à compter de la date du paiement

France & UE	Hors UE
❑ 90 € TTC	❑ 88,15 € HT

❑ Version duo

Version imprimée
+ version numérique

France & UE	Hors UE
❑ 155 € TTC	❑ 171,81 € HT*

Votre adhésion à la SEE

❑ Standard	❑ Retraité ❑ Enseignant ❑ Jeune actif (< 35 ans)	❑ Etudiant ❑ En recherche d'emploi
125 € TTC	65 € TTC	15 € TTC

+ Votre abonnement REE *(Tarif réservé aux adhérents, version papier)*

❑ France & UE : 63 € TTC ❑ Hors UE : 81,70 € HT*

TVA de la revue REE : 2,1 %. Adhésion collective possible via des conventions de partenariat - Contactez-nous à : sg@see.asso.fr
** Prix final incluant des frais de transports de 20 €*

Adresse de livraison

Nom* :

Prénom* :

Adresse* :

Code postal* : Pays* :

Ville* :

Tél.* :

e-mail* :

**Obligatoire*

Votre réglement

Je règle la somme de ____ €

par ❑ Chèque à l'ordre de la SEE
❑ Virement après réception de la facture
❑ Carte bancaire (Visa, Eurocard/Mastercard)

N° Carte

Date de validité ____ N° cryptogramme ____ *(3 derniers chiffres au dos de la carte)*

Adresse de facturation *(Si différente)*

❑ Je joins le bon de commande administratif N° ____ et je désire recevoir une facture au nom de mon employeur pour paiement à réception

Raison sociale de l'employeur :

Service : Activité (facultatif) :

Adresse :

Code postal : Ville :

Pays : N° TVA :

N° TVA intracommunautaire : obligatoire pour règlement HT en UE hors de France

e-mail* :

Date* ____ Signature* et cachet si il y a lieu :

**Obligatoire*

BULLETIN À COMPLÉTER ET RENVOYER À : SEE - 17 rue de l'Amiral Hamelin - 75116 Paris - France
Tél. +33(0)1 56 90 37 17 - abo@see.asso.fr

ABONNEMENT PLUS RAPIDE : www.see.asso.fr

see since 1883

❑ Je consens à recevoir les autres diffusions de la SEE & de ses activités (congrès, soirées débats, revues, etc.) qui sont extérieures aux diffusions liées à mon abonnement.

Commandez les numéros de La Revue 3E.I

Les archives de la 3EI, pour partager l'enseignement du génie électrotechnique et de l'électronique industrielle

❑ **N°95**/2019

Innovation du GE dans les transports

❑ **N°96**/2019

Réseaux à faibles consommation et longue portée

❑ **N°97**/2019

Capteurs et applications médicales

❑ **N°98**/2019

Projets étudiants

❑ **N°99**/2020

Véhicule électrifié, avenir de l'automobile ?

❑ **N°100**/2020

Véhicule électrifié, avenir de l'automobile ? (suite)

❑ **N°101**/2020

Idées de projets et enseignement à distance

❑ **N°102**/2020

Idée de projets et enseignements à distance

❑ **N°103**/2021

Enseignement du Génie Electrique à distance

❑ **N°104**/2021

Ampère 200 ans

❑ **N°105**/2021

Avion et électricité, vers une baisse des émissions de gaz à effet de serre ?

❑ **N°106**/2021

Petit voyage au pays de la robotique

La 3EI est une publication trimestrielle de la SEE. Vous pouvez commander les numéros parus dans la limite des stocks disponibles en remplissant le formulaire ci-dessous. Liste complète des archives disponible sur le site **www.see.asso.fr**

Tarif unique au numéro Prix unitaire TTC (TVA 2,10 %)	
France & UE*	Hors UE*
12 €	**22,21€**

**Frais de livraison inclus*

❑ **Je commande :**

N° : ____ Année : ____ N° : ____ Année : ____ N° : ____ Année : ____

N° : ____ Année : ____ N° : ____ Année : ____ N° : ____ Année : ____

N° : ____ Année : ____ N° : ____ Année : ____ N° : ____ Année : ____

N° : ____ Année : ____ N° : ____ Année : ____ N° : ____ Année : ____

Adresse de livraison

Nom :

Prénom :

Société :

N° TVA* :

Adresse :

Code postal : Pays :

Ville :

Tél. :

e-mail :

**Obligatoire pour les pays de l'UE*

Votre réglement

Je règle la somme totale de ____ € *(TVA 2,10 %)*

par ❑ Chèque à l'ordre de la SEE

❑ Virement après réception de la facture

❑ Carte bancaire (Visa, Eurocard/Mastercard)

N° Carte ____

Date de validité ____ N° cryptogramme ____ *(3 derniers chiffres au dos de la carte)*

Date* ____ Signature* et cachet si il y a lieu :

**Obligatoire*

BULLETIN À COMPLÉTER ET RENVOYER À : SEE - 17 rue de l'Amiral Hamelin - 75116 Paris - France
Tél. +33(0)1 56 90 37 17 - abo@see.asso.fr

❑ Je consens à recevoir les autres diffusions de la SEE & de ses activités (congrès, soirées débats, revues, etc.) qui sont extérieures aux diffusions liées à mon abonnement.

Conformément aux dispositions légales et réglementaires en matière de données personnelles, les informations recueillies sur ce formulaire sont enregistrées dans un fichier informatisé par la SEE (Société de l'électricité, de l'électronique et des technologies de l'information et de la communication) pour la mise en place et le suivi de l'abonnement souscrit ainsi que pour l'envoi de courriers, e-mails de réabonnements. Elles sont conservées et sont destinées à être utilisées par la SEE et les prestataires techniques de la SEE afin de permettre la bonne réception du magazine et d'assurer le service client. Vous pouvez exercer votre droit d'accès aux données vous concernant par courrier : SEE - Service abonnements 17 rue de l'Amiral Hamelin 75116 Paris ou par le formulaire de contact du site web : www.see.asso.fr. Offre valable du 01/01/2022 au 30/09/2022 dans la limite des quantités disponibles.

see since 1883